Strict Minimizers For Geometric Optimization Supplementary Material

Zohar Levi [*]
New York University

Denis Zorin [†]
New York University

1 Efficient One-Ring L_∞ Optimization

In 2D and for field-deviation distortion, or distortion measures with similar properties, the one-ring optimization problems can be solved efficiently, due to their special properties. We describe a specialized function that we use, with detailed analysis.

Equation of the curve of constant distortion. Suppose p_i, p_2, p are three vertices of a triangle T^p (in a local 2D coordinate system); q_1 and q_2 are fixed parametric-plane images of p_1 and p_2; and $q = [u, v]$ is the vertex p that is allowed to move, forming a triangle T^q.

Proposition 1. *The set of positions of point q for which the distortion is equal to a given k^2 is either empty, or a circle. The center of the circle does not depend on k.*

Proof. Define $v_2 = p_2 - p_1$, $v_3 = v_3 - p_1$, $s_2 = q_2 - q_1$, $x = q - q_1$. Let V be the matrix with columns v_1 and v_2, and

$$V^{-1} = \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix}$$

is its inverse, with rows w_1^T and w_2^T. Then, the Jacobian of the linear map mapping of T to T^q satisfies $JV = [s_2 x]$, i.e., can be written as

$$J = [s_2 x] \begin{bmatrix} w_1^T \\ w_2^T \end{bmatrix}$$

or in other words

$$J = s_2 w_1^T + x w_2^T$$

(sum of two vector outer products). Now we expand the ARAP distortion of the mapping

$$D(q) = \mathrm{tr}(J - R)^T (J - R) = \mathrm{tr}(J^T J) - 2\mathrm{tr}(R^T J) + 2$$

Using $\mathrm{tr}(ab^T) = a^T b$ formula for the trace of an outer product, we get the following explicit equation for $D(q) = k^2$:

$$w_2^2 x^2 + 2((w_1 \cdot w_2)s_2 - Rw_2) \cdot x + s_2^2 w_1^2 + 2s_2 \cdot Rw_1 + 2 = k^2$$

Introducing

$$\begin{aligned} A &= w_2^2 \\ \tilde{b} &= -\frac{1}{A}((w_1 \cdot w_2)s_2 - Rw_2) \\ C &= \frac{1}{A}(s_2^2 w_1^2 + 2s_2 \cdot Rw_1 + 2) \end{aligned}$$

we get:

$$x^2 - 2\tilde{b} \cdot x + C = \frac{k^2}{A} \quad .$$

[*]e-mail: zohar@cs.nyu.edu
[†]e-mail: dzorin@cs.nyu.edu

Substituting $x = q - q_1$, and defining $b = q_1 + \tilde{b}$, and $d = \tilde{b}^2 - C$, we get the equation for the equal distortion curve:

$$\|q - b\|^2 = \frac{k^2}{A} + d$$

The distortion itself is given by $D(q) = A(\|q - b\|^2 - d)$. This is an equation of a circle with center b which does not depend on k, and radius $\frac{k^2}{A} + d$. d, which is always negative, defines the minimal distortion a triangle can have in the parametric domain (e.g., if $|q_1 - q_2| = |p_1 - p_2|$, then it is zero, but in general it is not).

□

Cases of minimum. We solve the problem of minimizing distortion in a 1-ring of a vertex, with the boundary of the ring fixed, and only parametric coordinates $q = [u, v]$ of the vertex changing. This can be formulated as

$$k \to \min, \text{ subject to } D_i(q) < k^2 \tag{1}$$

where $i = 1 \ldots N$ is the index of the triangle incident at a vertex mapped to q. The geometric notion of distortion circles allows to define the space of feasible solutions of the constraints above in 3D space $(q, k) = (u, v, k)$. This is a set of circular cones, where each cone has a rounded apex, a vertical axis (along k), a cone angle defined by the parameter A_i, and a cone apex at height k_i, which is defined by $k_i^2/A_i + d_i = 0$, i.e., when the circle radius becomes zero.

The feasible domain is the intersection of feasible cones for all triangles T_i, $i = N$. This is a convex 3D domain with piecewise smooth boundary, and the solution of the optimization problem is the lowest point point of this domain. As the boundary is made of pieces of circular cones, there are several possibilities for the lowest point, which has to be on the boundary of the feasible region. Suppose q^* is the optimal solution, and k^* is the corresponding value of distortion.

Note that if two cones coincide, then there is a redundant constraint (that does not contribute to the intersection), so we assume that no two cones coincide (we check in the beginning and eliminate all redundancy). Then, any two cones in the set either do not intersect, have a curve intersection, or a point intersection. If the intersection is a point, it must be the apex of the cone.

Proposition 2. *The solution q^*, k^* of the problem* (1) *is at a vertex (an intersection of edges) or an edge of the boundary of the feasible domain, or at a cone apex.*

By linearity of the function (the objective function is linear: $k \to min$), the solution cannot be in the interior of the feasible domain.

The faces of the boundary of the feasible domain are 2D pieces of cone surfaces. Clearly, the minimum cannot be in the interior of a boundary face, unless it is an apex, because we can always go down in k towards the apex, while staying on the face. Algebraically, being on a face means $D_i(q) < (k^*)^2$ (strictly less) for all but one

$i = j$, and not being at the apex means that $\nabla_q D_j(q) \neq 0$, i.e., there is a direction in (u, v) plane decreasing $D_j(q)$. For sufficiently small change in q, all other $D_i(q)$ remain less than k, thus we can decrease k.

This leaves 2 cases: Curved edges of the feasible domain, where two cones intersect; and vertices of the domain, where three cones intersect, or an intersection at a cone apex. Note that in degenerate cases, three or more cones can intersect on a curve, and four or more cones can intersect at points. But in each such set of cones we can find a minimal set of two (for edges) and three (for vertices) defining it.

Note that this means that a simplex-type algorithm can be used to solve the problem. As the number of constraints is small, we describe a brute-force algorithm, that simply checks all vertices and edges.

Edge minimum. Each edge is defined by two non-coinciding cones (A_i, b_i, d_i) and (A_j, b_j, d_j), which cannot share an apex (otherwise they either coincide, or only intersect at the apex), i.e., $b_i \neq b_j$. The intersection of two distinct cones is a quadratic curve or a line, which has the lowest point. Consider the plane $k = k^*$, passing through this point. The cross-sections of both cones are circles, with a single intersection point (as this is the lowest point of the intersection), i.e., tangent circles. As the tangent of a circle $D(q) = k^2$ is $\nabla_q D(q)^\perp$, we have the following system of equations defining the edge minimum (we cannot just compare vectors, they might have different length):

$$\nabla_q D_i(q) \cdot \nabla_q D_j(q)^\perp = 0,\ D_i(q) - D_j(q) = 0 \tag{2}$$

At first sight, this may appear to be a system of two quadratic equations; however, one can see that the first equation is actually *linear*: $\nabla_q D(q) = 2A(q - b_i)$, and the right-hand side of the first equation is $4A_iA_j(q - b_i) \cdot (q^\perp - b_j^\perp) = 4A_iA_j(b_j \cdot b_j^\perp + (b_i - b_j)^\perp \cdot q)$, using $q^\perp \cdot q = 0$, thus, the problem reduces to solving the following system:

$$\begin{aligned} b_j \cdot b_j^\perp + (b_i - b_j)^\perp \cdot q &= 0,\ A_i(\|q - b_i\|^2 - d_i) \\ &= A_j(\|q - b_j\|^2 - d_j). \end{aligned} \tag{3}$$

Vertex minimum. A vertex of the feasible domain is determined by 3 non-coinciding cones $D_i = k^2$, $D_j = k^2$ and $D_k = k^2$, which solves the system

$$D_i(q) - D_j(q) = 0,\ D_i(q) - D_k(q) = 0. \tag{4}$$

Meaning, two equations of the same form as the second equation in (3). Again, it may appear that this is a system of two quadratic equations in two variables. However, we observe that these equations are of a special form. Specifically, the quadratic terms are $(A_i - A_j)q^2$ and $(A_i - A_k)^2 q^2$. If both are nonzero, one can be easily eliminated (by scaling one - we can scale the cones - and subtracting from the other, to get rid of quadratic terms), and one equation replaced with a linear one. Thus, in this case, we also have the problem reduced to the problem of solving a system of a linear and a quadratic equation.

Summary of the algorithm. The brute-force algorithm proceeds as follows:

1. Compute A_i, b_i, d_i for all T_i.
2. Apex candidates: For all T_i, compute $q_i^* = b_i$, and $(k_i^*)^2 = D_i(b_i) = -A_i d_i$.
3. For all pairs (T_i, T_j), solve (2) to obtain q_{ij}^*. The result may be an empty set, one solution, or two solutions.
4. For all triples (T_i, T_j, T_k), Solve (4) to obtain q_{ijk}^*.
5. In the set of candidate solutions

$$\begin{array}{l} \{(q_i^*, k_i^*) i = 1 \ldots N\} \ \cup \\ \{(q_{ij}^*, k_{ij}^*) i, j = 1 \ldots N, j > i\} \ \cup \\ \{(q_{ijk}^*, k_{ijk}^*) i, j = 1 \ldots N, k > j > i\} \end{array}$$

   pick the one with lowest value of feasible k.

Performance. Note that every step amounts to computing the coefficients and solving a set of quadratic equations in one variable. The total number of equations to be solved in the brute-force algorithm is $N + N(N-1)/2 + N(N-1)(N-2)/6$, i.e. 40 for the average valence 6. Note that this grows cubically, but as vertices of valence higher than 10 are very rare, this maintains good performance for almost all meshes.

An obvious optimization, which is likely to make the algorithm near-instantaneous (cost of several quadratic equations), is to build a simplex-like algorithm on top of this procedure. Our guess is that one can expect another factor of 5 improvement for valence 6, but for higher valences it can be more significant. We leave this as a future research direction.

Standard form of the equations. We can reduce both cases requiring solving a system of two equations to the form:

$$D_1 q^2 + B_1 \cdot q + C_1 = 0,\ D_2 q^2 + B_2 \cdot q + C_2 = 0, \tag{5}$$

where the coefficients are defined as follows.

1. For edges

$$\begin{aligned} D_1 &= 0 \\ B_1 &= (b_i - b_j)^\perp \\ C_1 &= b_j \cdot b_j^\perp \\ D_2 &= A_i - A_j \\ B_2 &= -2(A_i b_i - A_j b_j) \\ C_2 &= A_i(b_i^2 - d_i) - A_j(b_j^2 - d_j) \end{aligned}$$

2. For vertices, both equations are of the same form as the second equation for edges.

It remains to consider various cases of solving system (5):

1. $D_1 D_2 \neq 0$, $c = (\frac{B_1}{D_1} - \frac{B_2}{D_2}) \neq 0$. Then we replace the first equation with linear:

$$c \cdot q + f = 0,\ D_2 q^2 + B_2 \cdot q + C_2 = 0.$$

   where $f = (\frac{C_1}{D_1} - \frac{C_2}{D_2})$. The solutions of the first equation have the form $c^\perp t - fc/c^2$, where t is a scalar, resulting in quadratic equation

$$D_2(c^\perp t - f)^2 + B_2 \cdot (c^\perp t - f) + C_2 = 0$$

   equivalently $(D_2 c^2) t^2 + (2D_2 f + B_2) \cdot c^\perp t + (C_2 - B_2 \cdot f + D_2 f^2) = 0$.

2. $D_1 D_2 \neq 0$, $c = (\frac{B_1}{D_1} - \frac{B_2}{D_2}) = 0$. In this case, the equations are either incompatible, if f defined in the previous item is not zero, or coincide (which is not possible by assumption of excluding redundant cones), neither case is relevant.

3. $D_1 D_2 = 0$, but $D_1 \neq 0$ or $D_2 \neq 0$. If $D_1 = 0$, we define $c = B_1$, $f = C_1$, and reduce the problem to the first case, after replacement of one quadratic equation with linear. The other case $D_2 = 0$, $D_1 \neq 0$ is treated similarly.
4. $D_1 = D_2 = 0$. A system of two linear equations, solved in the usual way. If the lines coincide, the solutions are discarded.