An Optimal Triangle Projector with Prescribed Area and Orientation, Application to Position-Based Dynamics

Carlos Arango Duque*, Adrien Bartoli

EnCoV, IGT, Institut Pascal, UMR6602 CNRS Université Clermont Auvergne

Abstract

The vast majority of mesh-based modelling applications iteratively transform the mesh vertices under prescribed geometric conditions. This occurs in particular in methods cycling through the constraint set such as Position-Based Dynamics (PBD). A common case is the approximate local area preservation of triangular 2D meshes under external editing constraints. At the constraint level, this yields the nonconvex optimal triangle projection under prescribed area problem, for which there does not currently exist a direct solution method. In current PBD implementations, the area preservation constraint is linearised. The solution comes out through the iterations, without a guarantee of optimality, and the process may fail for degenerate inputs where the vertices are colinear or colocated. We propose a closed-form solution method and its numerically robust algebraic implementation. Our method handles degenerate inputs through a two-case analysis of the problem's generic ambiguities. We show in a series of experiments in area-based 2D mesh editing that using optimal projection in place of area constraint linearisation in PBD speeds up and stabilises convergence.

Keywords: triangle, optimal projection, area preservation, orientation preservation, mesh editing, PBD

1. Introduction

A key mechanism in many mesh-based modelling applications is to transform the mesh vertices to meet prescribed geometric conditions. For example, triangular mesh smoothing may be achieved by moving the vertices of each triangle by a specifically

*Corresponding author

Email address: Carlos.ARANGO_DUQUE@uca.fr (Carlos Arango Duque)

designed two-step stretching-shrinking transformation [1] or by iteratively applying a local smoothing transformation [2]. Another example is 3D volumetric model deformation, where realism is improved by preserving the volume of the mesh's tetrahedrons [3, 4]. Position-Based Dynamics (PBD) is a widely used simulation technique that directly manipulates the vertex positions of object meshes. It can model various object behaviours such as rigid body, soft body and fluids [5]. Due to its simplicity, robustness and speed, PBD has become very popular in computer graphics and in the video-game industry. In general terms, PBD updates the vertex positions through simple integration of the external forces. These positions are then directly subjected to a series of constraint equations handled one at a time. If the obtained projection minimises the vertex displacement then it is qualified as optimal. For example, the projection for vertex distance preservation is a simple problem, which was solved optimally [6, 7]. The constraints simulate a wide range of effects like stretching, bending, collision, area and volume conservation [7]. Over the years, improvements have been proposed to the original formulation of PBD. These include new bending constraints from simple geometric principles [8, 9], stability improvement by geometric stiffness [10] and faster convergence by constraint reordering [11].

Mesh editing uses a priori chosen fixed vertices and moving vertices which should respect constraints. In 2D triangular mesh editing, local area preservation is a widely used constraint. The mesh is deformed until the triangle-wise area variation is minimised. Enforcing this constraint leads to the Optimal Triangle Projection with Prescribed Area problem (OTPPA), which we will formally define shortly. OTPPA is a difficult problem and has not yet been given a closed-form solution in the literature, contrarily to optimal vertex distance preservation. We formally define OTPPA as follows. We define the vertices v_a, v_b and v_c of a triangle in a 2D space as a 6D vector $\mathbf{v}$ with:

$$\mathbf{v} = [v_a, v_b, v_c]^\top = [x_a, y_a, x_b, y_b, x_c, y_c]^\top \in \mathbb{R}^6. \tag{1}$$

We denote the input triangle $\tilde{\mathbf{v}} = [\tilde{x}_a, \tilde{y}_a, \tilde{x}_b, \tilde{y}_b, \tilde{x}_c, \tilde{y}_c]^\top$ and the prescribed area A_o. We assume $A_o > 0$ out of practical considerations[1]. The general OTPPA problem is stated as:

$$\min_{\mathbf{v}\in\mathbb{R}^6} \mathscr{C}(\mathbf{v}) \quad \text{s.t.} \quad f(\mathbf{v}) = 0, \tag{2}$$

where $\mathscr{C}(\mathbf{v})$ is the least-squares *displacement cost*:

$$\mathscr{C}(\mathbf{v}) = \|\mathbf{v} - \tilde{\mathbf{v}}\|^2, \tag{3}$$

[1] $A_o = 0$ implies that the resulting vertices are colinear, in which case they are simply given by the orthogonal projection of the input vertices onto a best-fit least-squares line to the input vertices.

and $f(\mathbf{v})$ is the nonconvex *area preservation constraint*, defined from the triangle area function $A(\mathbf{v})$ as:

$$f(\mathbf{v}) = A(\mathbf{v}) - A_o. \tag{4}$$

Areas are positive quantities. This means that when we calculate the area of a triangle using its vertices, we obtain a positive value regardless of their orientation. In this formulation, the area constraint is the difference of two non-negative values. That is, the area of the triangle is constrained but not its orientation. For instance, $\mathbf{v}$ could be mirrored or two vertices could be swapped and the area constraint would still be satisfied. This could result in undesired triangle inversions in mesh editing. We thus introduce a related problem that additionally constrains the triangle orientation. We first define the triangle area function as $A(\mathbf{v}) = |A^*(\mathbf{v})|$, where $A^*(\mathbf{v})$ is the *signed area* given by the shoelace formula:

$$A^*(\mathbf{v}) = \frac{(x_a - x_c)(y_b - y_a) - (x_a - x_b)(y_c - y_a)}{2}. \tag{5}$$

The signed area is more informative than the area. Specifically, $\mathrm{sign}(A^*(\mathbf{v}))$ gives the triangle orientation. We use this property to define the additional orientation constraint. This leads to the Optimal Triangle Projection with Prescribed Area and Orientation problem (OTPPAO), stated as:

$$\min_{\mathbf{v}\in\mathbb{R}^6} \mathscr{C}(\mathbf{v}) \quad \text{s.t.} \quad f(\mathbf{v}) = 0 \quad \text{and} \quad g(\mathbf{v}) = 0, \tag{6}$$

where $g(\mathbf{v})$ is the *orientation preservation constraint*:

$$g(\mathbf{v}) = \mathrm{sign}(A^*(\mathbf{v})) - s, \tag{7}$$

and $s \in \{-1, 1\}$ specifies the prescribed orientation. The value chosen for s depends on the application. For instance, in PBD, one would choose the orientation of the reference mesh triangle, while another possibility would be to preserve the orientation of the input triangle by setting $s = \mathrm{sign}(A^*(\tilde{\mathbf{v}}))$.

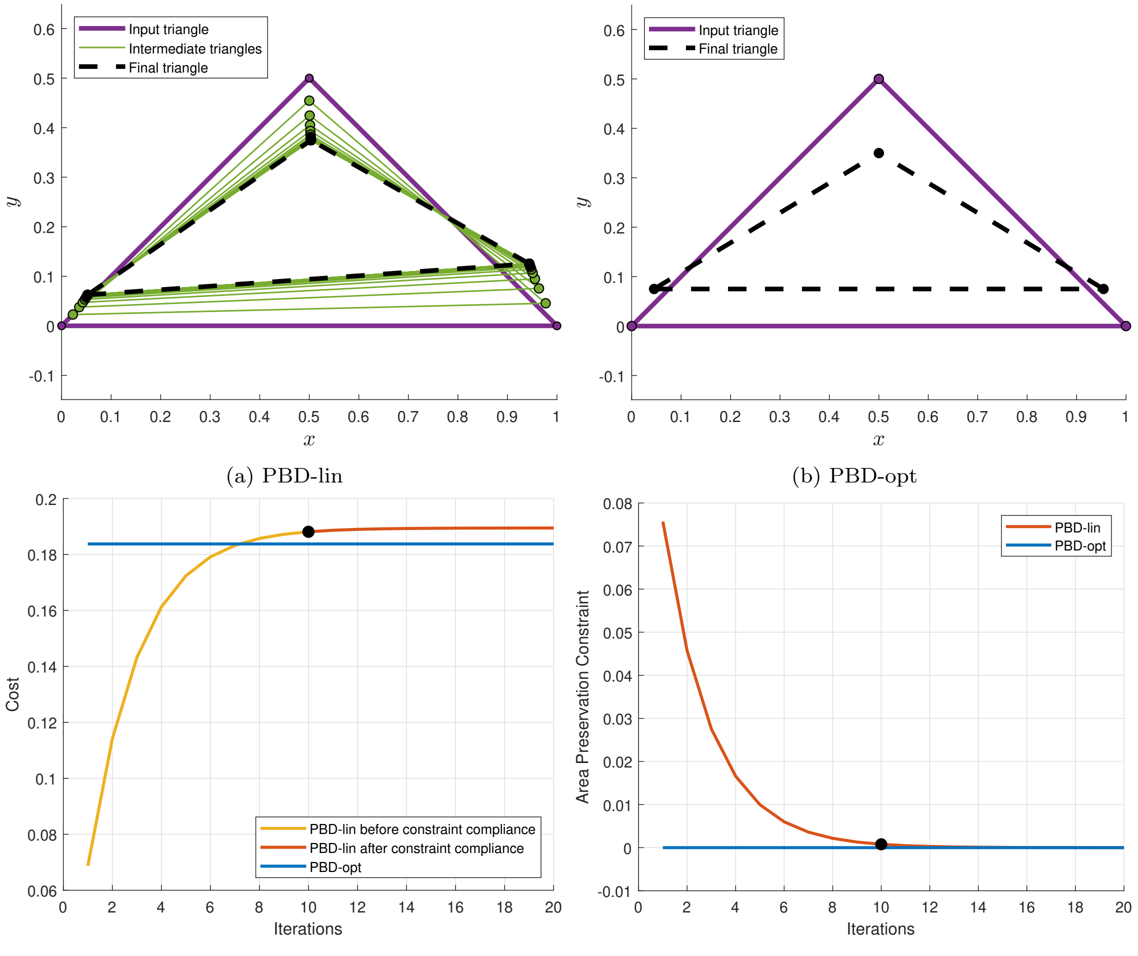
For both OTPPA and OTPPAO, current PBD implementations linearise the area preservation constraint, resulting in suboptimal projection. For OTPPAO, they also check and enforce the orientation constraint a posteriori in the inner optimisation loop. There also exist implementations based on non-linear optimisation, namely Sequential Quadratic Programming (SQP), which may result in faster convergence [12]. Similarly to linearisation, these methods may fail to find the global minimum. This will be critical when the problem is close to a degenerate configuration admitting several solutions. In addition, any iterative method will require an unpredictable number of iterations to converge, rendering the actual computation time difficult to predict. Furthermore, they degenerate for inputs where the vertices are colinear or colocated, whereas a reliable solution should handle any input. We can thus expect an optimal projection to improve PBD convergence compared to linearisation and iterative methods.

We propose a closed-form method to OTPPAO and OTPPA. Our method for OTPPAO handles degenerate inputs through a two-case analysis, guaranteeing it to find the optimal solution and returning multiple optimal solutions for ambiguous inputs. Our method for OTPPA directly relies on OTPPAO and shares the same features. Our method derivation, although thoroughly detailed and relying on some complex steps, results in a simple algebraic procedure which can be readily implemented in any programming language. We use our closed-form method to implement PBD, hence dubbed PBD-opt, for mesh editing and compare its performance with respect to the existing PBD implementation with linearisation, dubbed PBD-lin. To illustrate our proposal, we present a one triangle toy example in Figure 1 in which we wish to resize the triangle to half its initial area. PBD-lin takes several iterations to reach the prescribed area, whereas PBD-opt achieves it directly. The cost evolution shows that PBD-lin starts with a lower cost, but by the time it complies with the area constraint, it reaches a larger cost than PBD-opt, indicating convergence to a local suboptimal minimum.

This paper has two parts. In the first part, we derive our closed-form methods. We show that they deal with generic ambiguities. We then implement our methods as numerically robust algebraic procedures. In the second part, we embed our algebraic procedure for OTTPAO in PBD to form an implementation of PBD-opt. We compare its performance in convergence speed and stability with respect to the existing PBD-lin in a series of experiments and present some use-cases. We finally give a complementary section where we present our solution to OTPPA and specialise our methods to cases where one or two triangle vertices are fixed, which is typically applicable to triangles at the domain boundary in mesh editing.

2. Optimal Triangle Projection with Prescribed Area and Orientation

The derivation of our closed-form method to OTPPAO starts by combining the two constraints into a single one related to both triangle area and orientation. We then construct the Lagrangian which leads to a nonconvex problem, which we handle with two cases. We distinguish and geometrically interpret the two cases based on the input vertices. In the first case, we reformulate the problem as a depressed quartic equation and solve it analytically. In the second case, we reformulate the



(a) PBD-lin

(b) PBD-opt

(c) Evolution of cost

(d) Evolution of area constraint

Figure 1: Method comparison on a one triangle toy example. (a), PBD-lin resizes the initial triangle (purple) into smaller intermediary triangles (green) until it reaches a triangle with the prescribed area (black dashed). (b), PBD-opt directly reaches the prescribed area. (c), Comparison of the evolution of the cost of PBD-lin and the fixed cost of PBD-opt. (d), Comparison of the evolution of the prescribed area constraint of PBD-lin and the fixed area of PBD-opt. The cost of PBD-opt (blue) is constant as it gives a direct solution. The cost of PBD-lin is lower during the first iterations (yellow) but the area constraint tolerance $|(A(\mathbf{v}) - A_o| \leq 10^{-3}$ is not yet fulfilled. By the time it reaches the prescribed area (black dot), the cost of PBD-lin (red) has become larger than the one of PBD-opt. Furthermore, after 20 iterations the area constraint for PBD-lin is $4.802 \cdot 10^{-6}$ compared to $8.327 \cdot 10^{-17}$ in PBD-opt.

problem as a series of homogeneous equations and find its null space. Based on these procedures, we develop a numerically robust algebraic implementation and show the results in a series of illustrative examples.

2.1. Single Constraint Reformulation

We reformulate OTPPAO by merging the area and orientation constraints into a single equivalent constraint:

$$f^*(\mathbf{v}) = 0 \quad \text{with} \quad f^*(\mathbf{v}) = sA^*(\mathbf{v}) - A_o. \tag{8}$$

We have $(f(\mathbf{v}) = 0) \wedge (g(\mathbf{v}) = 0) \Leftrightarrow f^*(\mathbf{v}) = 0$. The forward implication is obtained by rewriting $f(\mathbf{v}) = 0$ as $\text{sign}(A^*(\mathbf{v}))A^*(\mathbf{v}) - A_o = 0$ and substituting $s = \text{sign}(A^*(\mathbf{v}))$, as obtained from $g(\mathbf{v}) = 0$, directly giving $f^*(\mathbf{v}) = 0$. The reverse implication is obtained by rewriting $f^*(\mathbf{v}) = 0$ as $sA^*(\mathbf{v}) = A_o$, whose absolute value gives $f(\mathbf{v}) = 0$ and whose sign gives $g(\mathbf{v}) = 0$. With this new constraint, we reformulate OTPPAO as:

$$\min_{\mathbf{v}\in\mathbb{R}^6} \mathscr{C}(\mathbf{v}) \quad \text{s.t.} \quad f^*(\mathbf{v}) = 0. \tag{9}$$

This reformulation increases compactness but, more importantly, in contrast to the previous area constraint, the new constraint does not involve an absolute value. More specifically, $f^*(\mathbf{v})$ is a nonconvex but smooth function of $\mathbf{v}$, meaning that a Lagrangian formulation can now be safely constructed.

2.2. Lagrangian Formulation

The Lagrangian of the OTPPAO problem (9) is:

$$\mathscr{L}(\mathbf{v}, \lambda) = \mathscr{C}(\mathbf{v}) + \lambda f^*(\mathbf{v}), \tag{10}$$

where λ is the Lagrange multiplier. Setting the gradient to nought we obtain:

$$\frac{\partial \mathscr{L}}{\partial \lambda} = f^*(\mathbf{v}) = sA^*(\mathbf{v}) - A_o = 0 \tag{11}$$

$$\frac{\partial \mathscr{L}}{\partial \mathbf{v}} = \frac{\partial \mathscr{C}(\mathbf{v})}{\partial \mathbf{v}} + \lambda \frac{\partial f^*(\mathbf{v})}{\partial \mathbf{v}} = 2(\mathbf{v} - \tilde{\mathbf{v}}) + s\lambda \frac{\partial A^*(\mathbf{v})}{\partial \mathbf{v}} = 0. \tag{12}$$

Expanding $\frac{\partial \mathscr{L}}{\partial \mathbf{v}}$, we obtain the following six equations:

$$\begin{aligned}
\frac{\partial \mathscr{L}}{\partial x_a} &= 2(x_a - \tilde{x}_a) + s\frac{\lambda}{2}(y_b - y_c) = 0 \\
\frac{\partial \mathscr{L}}{\partial y_a} &= 2(y_a - \tilde{y}_a) + s\frac{\lambda}{2}(x_c - x_b) = 0 \\
\frac{\partial \mathscr{L}}{\partial x_b} &= 2(x_b - \tilde{x}_b) + s\frac{\lambda}{2}(y_c - y_a) = 0 \\
\frac{\partial \mathscr{L}}{\partial y_b} &= 2(y_b - \tilde{y}_b) + s\frac{\lambda}{2}(x_a - x_c) = 0 \\
\frac{\partial \mathscr{L}}{\partial x_c} &= 2(x_c - \tilde{x}_c) + s\frac{\lambda}{2}(y_a - y_b) = 0 \\
\frac{\partial \mathscr{L}}{\partial y_c} &= 2(y_c - \tilde{y}_c) + s\frac{\lambda}{2}(x_b - x_a) = 0.
\end{aligned}$$

We rewrite these equations in matrix form as:

$$X\mathbf{v} = \tilde{\mathbf{v}}, \tag{13}$$

where $X \in \mathbb{R}^{6\times 6}$ is given by:

$$X = \begin{bmatrix} 1 & 0 & 0 & s\lambda/4 & 0 & -s\lambda/4 \\ 0 & 1 & -s\lambda/4 & 0 & s\lambda/4 & 0 \\ 0 & -s\lambda/4 & 1 & 0 & 0 & s\lambda/4 \\ s\lambda/4 & 0 & 0 & 1 & -s\lambda/4 & 0 \\ 0 & s\lambda/4 & 0 & -s\lambda/4 & 1 & 0 \\ -s\lambda/4 & 0 & s\lambda/4 & 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

2.3. Solving with Two Cases

We want to solve for $\mathbf{v}$ from equation (14). We first check the invertibility of X from its determinant:

$$\det(X) = \frac{(3\lambda^2 - 16)^2}{256}. \tag{15}$$

We thus have:

$$\det(X) = 0 \quad \Leftrightarrow \quad |\lambda| = \lambda_o, \tag{16}$$

where λ_o correspond to the reciprocal area of a normalised equilateral triangle:

$$\lambda_o = \frac{4}{\sqrt{3}}. \tag{17}$$

We show in the next section that this special case is related to input vertices representing an equilateral triangle or being colocated. We thus solve system (14) with two cases. In Case I, which is the most general one, we have $|\lambda| \neq \lambda_o$. In Case II, we have $|\lambda| = \lambda_o$.

2.4. Geometrically Interpreting and Distinguishing the Two Cases

The problem setting is defined by the input vertices $\tilde{\mathbf{v}}$, the prescribed area A_o and orientation s. Most settings are solved by Case I and exceptions are handled by Case II. Both cases can be geometrically interpreted and distinguished from each other based on three criteria:

- **Linear deficiency of $\tilde{\mathbf{v}}$.** This is evaluated as the rank of matrix $M \in \mathbb{R}^{3\times 3}$ containing $\tilde{\mathbf{v}}$ in homogeneous coordinates as:

$$M \overset{\text{def}}{=} \begin{bmatrix} \tilde{x}_a & \tilde{y}_a & 1 \\ \tilde{x}_b & \tilde{y}_b & 1 \\ \tilde{x}_c & \tilde{y}_c & 1 \end{bmatrix}. \tag{18}$$

  For most configurations $\text{rank}(M) = 3$, which means that the vertices are not aligned and represent any given triangle whose area $A(\tilde{\mathbf{v}})$ is non-zero. In contrast, $\text{rank}(M) = 2$ means that the three vertices are colinear, in which case $A(\tilde{\mathbf{v}}) = 0$. Finally, $\text{rank}(M) = 1$ means that the three vertices are colocated and also implies $A(\tilde{\mathbf{v}}) = 0$.

- **Orientation change of $\tilde{\mathbf{v}}$.** This is evaluated by comparing the input vertices orientation $\text{sign}(A^*(\tilde{\mathbf{v}}))$ and the prescribed orientation s. When the orientation of the input vertices is preserved then $\text{sign}(A^*(\tilde{\mathbf{v}})) = s$. On the other hand, when the orientation of the input vertices is inverted then $\text{sign}(A^*(\tilde{\mathbf{v}})) = -s$.

- **Scale of $A(\tilde{\mathbf{v}})$ with respect to A_o.** This is only useful in the special case of an equilateral triangle with preserved orientation. It refers to whether the absolute value of the scaled input area $|zA^*(\tilde{\mathbf{v}})|$ is larger than, equal to or smaller than the prescribed area A_o for some $z \in \mathbb{R} > 0$.

The interpretation of Cases I and II with the above criteria is given in Table 1 and summarised by the following Proposition.

Proposition 1. *We define a problem setting as the input vertices $\tilde{\mathbf{v}}$, the prescribed area A_o and orientation s. Most settings fall in Case I and are then denoted S_o. Exceptions, handled with Case II, are:*

- *S_1: $\tilde{\mathbf{v}}$ is a single point*
- *S_2: $\tilde{\mathbf{v}}$ is an equilateral triangle and* $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = -s$
- *S_3: $\tilde{\mathbf{v}}$ is an equilateral triangle, $A(\tilde{\mathbf{v}})/4 \geq A_o$ and* $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$

The proof of Proposition 1 is based on the following five Lemmas.

Lemma 1. *$S_1 \iff A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = \lambda_o$.*

Lemma 2. *$S_2 \iff A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -\lambda_o$.*

Lemma 3. *$S_3 \Rightarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda \in \left\{\lambda_o, -\lambda_o + \sqrt{\frac{\lambda_o}{A_o}}, -\lambda_o - \sqrt{\frac{\lambda_o}{A_o}}\right\}$.*

Lemma 4. *$S_3 \Leftarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = \lambda_o$.*

Lemma 5. *Choosing $\lambda = \lambda_o$ leads to the optimal solution for S_3.*

The proofs of these Lemmas are given in Appendix A.

Proof of Proposition 1. We recall that Case I occurs for $|\lambda| \neq \lambda_o$ and Case II for $|\lambda| = \lambda_o$. Lemmas 1, 2 and 4 show that S_1, S_2 and S_3 are the only possible settings corresponding to $|\lambda| = \lambda_o$, hence possibly to Case II. This proves that Case I is the general case. Lemmas 1 and 2 then trivially prove that S_1 and S_2 are handled by Case II. Finally, Lemmas 3 and 5 prove that S_3 is also handled by Case II. □

2.5. Case I

Case I is the most general one. It occurs for $|\lambda| \neq \lambda_o$, equivalent to $\det(X) \neq 0$. From Proposition 1, we have $\operatorname{rank}(M) \geq 2$, in other words, at least one of the initial vertices $\tilde{\mathbf{v}}$ is different from the other two (except if the input is an equilateral triangle under the conditions of Proposition 1). We follow two steps. We first eliminate the vertices from the equations, which leads to a depressed quartic in λ. We then find the roots of this quartic using Ferrari's method and trivially solve for the vertices from the initial linear system (13).

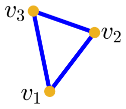
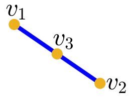
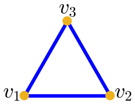
Case	Setting	Input	$\det(X)$	$\mathrm{rank}(M)$	$A(\tilde{\mathbf{v}})$	$\sigma^2(\tilde{\mathbf{v}})$	Number of solutions	$s\,\mathrm{sign}(A(\tilde{\mathbf{v}}))$
I	S_o		non zero	3	non zero	non zero	≤ 4	±1
				2	zero			
II	S_1	v_1=v_2=v_3	zero	1	zero	zero	∞	±1
I	S_o		non zero	3	$A(\tilde{\mathbf{v}})/4 \leq A_o$	non zero	≤ 4	1
II	S_3		zero		$A(\tilde{\mathbf{v}})/4 \geq A_o$		∞	1
II	S_2		zero		non zero		∞	-1

Table 1: Characteristics of Cases I and II and number of solutions.

2.5.1. Polynomial Reformulation

Our reformulation proceeds by expressing the vertices in $\tilde{\mathbf{v}}$ as a function of λ scaled by the determinant and substituting in the signed area constraint $f^*(\mathbf{v}) = 0$. We start by multiplying equation (13) by the adjugate X^* of X and obtain:

$$\det(X)\mathbf{v} = X^*\tilde{\mathbf{v}}, \tag{19}$$

where the adjugate is:

$$X^* = \frac{\delta}{256}Y = \frac{3\lambda^2 - 16}{256}\begin{bmatrix} \lambda^2 - 16 & 0 & \lambda^2 & 4s\lambda & \lambda^2 & -4s\lambda \\ 0 & \lambda^2 - 16 & -4s\lambda & \lambda^2 & 4s\lambda & \lambda^2 \\ \lambda^2 & -4s\lambda & \lambda^2 - 16 & 0 & \lambda^2 & 4s\lambda \\ 4s\lambda & \lambda^2 & 0 & \lambda^2 - 16 & -4s\lambda & \lambda^2 \\ \lambda^2 & 4s\lambda & \lambda^2 & -4s\lambda & \lambda^2 - 16 & 0 \\ -4s\lambda & \lambda^2 & 4s\lambda & \lambda^2 & 0 & \lambda^2 - 16 \end{bmatrix}, \tag{20}$$

with $\delta = 3\lambda^2 - 16$ and $Y \in \mathbb{R}^{6\times 6}$. We notice the following:

$$\det(X) = \frac{\delta^2}{256}. \tag{21}$$

We substitute equations (20) and (21) in equation (19) and obtain:

$$\delta\mathbf{v} = Y\tilde{\mathbf{v}}. \tag{22}$$

We observe that the signed area $A^*(\delta\mathbf{v}) = \delta^2 A^*(\mathbf{v})$. Thus, we calculate the signed area of both sides of equation (19) and obtain:

$$\delta^2 A^*(\mathbf{v}) = A^*(Y\tilde{\mathbf{v}}). \tag{23}$$

After some minor manipulations, we obtain:

$$A^*(Y\tilde{\mathbf{v}}) = a_2\lambda^2 + a_1\lambda + a_o, \tag{24}$$

where:

$$\begin{aligned} a_0 &= 128((\tilde{x}_a - \tilde{x}_c)(\tilde{y}_b - \tilde{y}_a) - (\tilde{x}_a - \tilde{x}_b)(\tilde{y}_c - \tilde{y}_a)) \\ a_1 &= -64s(\tilde{x}_a^2 + \tilde{x}_b^2 + \tilde{x}_c^2 + \tilde{y}_a^2 + \tilde{y}_b^2 + \tilde{y}_c^2 - \tilde{x}_a\tilde{x}_b - \tilde{x}_a\tilde{x}_c - \tilde{x}_b\tilde{x}_c - \tilde{y}_a\tilde{y}_b - \tilde{y}_a\tilde{y}_c - \tilde{y}_b\tilde{y}_c) \\ a_2 &= 24((\tilde{x}_a - \tilde{x}_c)(\tilde{y}_b - \tilde{y}_a) - (\tilde{x}_a - \tilde{x}_b)(\tilde{y}_c - \tilde{y}_a)). \end{aligned}$$

We can rewrite these coefficients more compactly. Concretely, a_0 and a_2 contain the signed area of the input vertices $A^*(\tilde{\mathbf{v}})$, as given by equation (5). Furthermore, a_1

is proportional to the sum of the variance of the x and y components of the input vertices, as $\sigma^2(\tilde{\mathbf{v}}) = \sigma_x^2(\tilde{\mathbf{v}}) + \sigma_y^2(\tilde{\mathbf{v}})$ where:

$$\sigma_x^2(\tilde{\mathbf{v}}) = \frac{(\tilde{x}_a - \bar{x})^2 + (\tilde{x}_b - \bar{x})^2 + (\tilde{x}_c - \bar{x})^2}{3}, \tag{25}$$

$$\sigma_y^2(\tilde{\mathbf{v}}) = \frac{(\tilde{y}_a - \bar{y})^2 + (\tilde{y}_b - \bar{y})^2 + (\tilde{y}_c - \bar{y})^2}{3}, \tag{26}$$

and $\bar{x}$ and $\bar{y}$ are the x and y components of the input triangle's centroid. We then have:

$$\frac{3}{2}\sigma^2(\tilde{\mathbf{v}}) = \tilde{x}_a^2 + \tilde{x}_b^2 + \tilde{x}_c^2 + \tilde{y}_a^2 + \tilde{y}_b^2 + \tilde{y}_c^2 - \tilde{x}_a\tilde{x}_b - \tilde{x}_a\tilde{x}_c - \tilde{x}_b\tilde{x}_c - \tilde{y}_a\tilde{y}_b - \tilde{y}_a\tilde{y}_c - \tilde{y}_b\tilde{y}_c, \tag{27}$$

and thus, $a_1 = -96s\sigma^2(\tilde{\mathbf{v}})$. We substitute equation (23) in the signed area constraint (8) multiplied by δ^2 and obtain:

$$sA^*(Y\tilde{\mathbf{v}}) - \delta^2 A_o = 0. \tag{28}$$

This way, the signed area only depends on the known initial vertices $\tilde{\mathbf{v}}$ and prescribed sign s. Because the signed area is quadratic in the vertices, and the vertices are quadratic rational in λ, the resulting equation is a quartic in λ:

$$9A_o\lambda^4 - 48(2A_o + sA^*(\tilde{\mathbf{v}}))\lambda^2 + 96\sigma^2(\tilde{\mathbf{v}})\lambda + 256(A_o - sA^*(\tilde{\mathbf{v}})) = 0. \tag{29}$$

This is a depressed quartic because it does not have a cubic term. We can thus rewrite it to the standard form by simply dividing by $9A_o$, giving:

$$\lambda^4 + p\lambda^2 + q\lambda + r = 0, \tag{30}$$

with:

$$p = -\frac{16(2A_o + sA^*(\tilde{\mathbf{v}}))}{3A_o} \tag{31}$$

$$q = \frac{32\sigma^2(\tilde{\mathbf{v}})}{3A_o} \tag{32}$$

$$r = \frac{256(A_o - sA^*(\tilde{\mathbf{v}}))}{9A_o}. \tag{33}$$

An important question is whether we can further simplify the depressed quartic by nullifying one of its coefficients. The possible actions lie in choosing the coordinate frame in which the vertices are expressed using a proper scaled Euclidean transformation. Coefficients p and r are linear combinations of the triangle areas $A^*(\tilde{\mathbf{v}})$ and A_o and are normalised by A_o. This means that they are scale, rotation and translation invariant, and thus cannot be cancelled. Coefficient q is proportional to the variance $\sigma^2(\tilde{\mathbf{v}})$ which is also rotation and translation invariant, and thus cannot be cancelled. Because it is normalised by the area, it is also scale invariant. Consequently, since λ is a root of this polynomial, it is also rotation, translation and scale invariant. Therefore, the depressed quartic cannot be simplified. The next step is to solve the depressed quartic.

2.5.2. Solution using Ferrari's Method

We have chosen Ferrari's method [13, 14] to solve the quartic equation[2]. The method details and proof may be found in Appendix C. We here give its main steps for the sake of completeness and for the construction of our numerically robust procedure in Section 2.8. We first extract the resolvent cubic for equation (30). We then use Cardano's formula to extract the real root α_o of the resolvent cubic as:

$$\alpha_o = \sqrt[3]{Q_2 + \sqrt{Q_1^3 + Q_2^2}} + \sqrt[3]{Q_2 - \sqrt{Q_1^3 + Q_2^2}} - \frac{p}{3} \tag{34}$$

where:

$$Q_1 = -\frac{p^2 + 12r}{36} \tag{35}$$

$$Q_2 = \frac{2p^3 - 72rp + 27q^2}{432}. \tag{36}$$

The expansion of Q_1 and Q_2 does not bring simplified expressions. We note that Proposition 1 implies $q \neq 0$, thus $\alpha_o \neq 0$. We finally use α_o to extract the roots of the depressed quartic as:

$$\lambda = \frac{s_1\sqrt{\alpha_o} + s_2\sqrt{-\left(p + \alpha_o + s_1\frac{q}{\sqrt{2\alpha_o}}\right)}}{\sqrt{2}}, \tag{37}$$

where $s_1, s_2 \in \{-1, 1\}$, leaving four possibilities, hence four roots. Substituting these roots in equation (13), we obtain four sets of vertices, at least one of which representing an optimal solution to OTPPAO.

[2] There are five main types of solution methods for a quartic equation. There does not seem to exist a consensus as to which one should be preferred in terms of stability [15, 16, 17].

2.6. Case II

Case II is a special case. It occurs for $|\lambda| = \lambda_o$, equivalent to $\det(X) = 0$. From Proposition 1, this means that the initial vertices $\tilde{\mathbf{v}}$ either are colocated as $\tilde{v}_a = \tilde{v}_b = \tilde{v}_c$ or represent equilateral triangles under the conditions of Proposition 1. We show that the problem is represented by translated homogeneous and linearly dependent equations. We find their null space and then a subset constrained by the prescribed area.

We translate the coordinate system to bring the input triangle's centroid to the origin as $\tilde{\mathbf{v}}' = \tilde{\mathbf{v}} - \bar{\mathbf{v}}$, which also translates the unknown vertices to $\mathbf{v}' = \mathbf{v} - \bar{\mathbf{v}}$. Substituting $|\lambda| = \lambda_o$ in matrix X, we obtain:

$$X = \begin{bmatrix} 1 & 0 & 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} \\ 0 & 1 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 \\ 0 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 1 & 0 & 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} \\ \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & 0 & 1 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 \\ 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 1 & 0 \\ -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & 0 & 1 \end{bmatrix}. \tag{38}$$

We have $\det(X) = 0$, as expected, independently of $\mathrm{sign}(\lambda)$. In addition, all 5×5 minors of X are zero and the leading 4×4 minor is non-zero:

$$\det\left(\begin{bmatrix} 1 & 0 & 0 & \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} \\ 0 & 1 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 \\ 0 & -\frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 1 & 0 \\ \frac{s\,\mathrm{sign}(\lambda)}{\sqrt{3}} & 0 & 0 & 1 \end{bmatrix}\right) = \frac{4}{9}. \tag{39}$$

This means that $\mathrm{rank}(X) = 4$. Thus, $X\mathbf{v}' = \tilde{\mathbf{v}}'$ is solvable if and only if $\tilde{\mathbf{v}}'$ lies in the column space $C(X)$. The column space can be calculated by factoring X into its Singular Value Decomposition (SVD) $X = U\Sigma U^\top$ (X is symmetric) and taking the first $\mathrm{rank}(X)$ columns of the unitary matrix U. For each value of $s\,\mathrm{sign}(\lambda)$, we have column spaces expressed as four-dimensional linear subspaces $\{\gamma_1\mathbf{u}_1^- + \gamma_2\mathbf{u}_2^- + \gamma_3\mathbf{u}_3^- + \gamma_4\mathbf{u}_4^-\}$ and $\{\gamma_1\mathbf{u}_1^+ + \gamma_2\mathbf{u}_2^+ + \gamma_3\mathbf{u}_3^+ + \gamma_4\mathbf{u}_4^+\}$ where $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \mathbb{R}$ and with bases $\mathbf{u}_1^-, \mathbf{u}_2^-, \mathbf{u}_3^-, \mathbf{u}_4^- \in \mathbb{R}^6$ and $\mathbf{u}_1^+, \mathbf{u}_2^+, \mathbf{u}_3^+, \mathbf{u}_4^+ \in \mathbb{R}^6$ such that:

$$\begin{bmatrix} \mathbf{u}_1^- & \mathbf{u}_2^- & \mathbf{u}_3^- & \mathbf{u}_4^- \end{bmatrix} = \begin{bmatrix} 0 & \lambda_o/4 & \lambda_o/4 & 0 \\ \lambda_o/4 & 0 & 0 & \lambda_o/4 \\ 1/2 & -\lambda_o/8 & \lambda_o/4 & 0 \\ -\lambda_o/8 & -1/2 & 0 & \lambda_o/4 \\ -1/2 & -\lambda_o/8 & \lambda_o/4 & 0 \\ -\lambda_o/8 & 1/2 & 0 & \lambda_o/4 \end{bmatrix}, \tag{40}$$

and:

$$\begin{bmatrix} \mathbf{u}_1^+ & \mathbf{u}_2^+ & \mathbf{u}_3^+ & \mathbf{u}_4^+ \end{bmatrix} = \begin{bmatrix} \lambda_o/4 & 0 & 0 & \lambda_o/4 \\ 0 & \lambda_o/4 & \lambda_o/4 & 0 \\ -\lambda_o/8 & -1/2 & 0 & \lambda_o/4 \\ 1/2 & -\lambda_o/8 & \lambda_o/4 & 0 \\ -\lambda_o/8 & 1/2 & 0 & \lambda_o/4 \\ -1/2 & -\lambda_o/8 & \lambda_o/4 & 0 \end{bmatrix}. \tag{41}$$

We have that $\mathbf{u}_1^-, \mathbf{u}_2^-, \mathbf{u}_1^+$ and $\mathbf{u}_2^+$ represent centred equilateral triangles of the same area of $\frac{1}{\lambda_o}$ with orientation $s\,\mathrm{sign}(\lambda)$ and we have that $\mathbf{u}_3^-, \mathbf{u}_4^-, \mathbf{u}_3^+$ and $\mathbf{u}_4^+$ represent sets of colocated points. The linear combinations $\gamma_1\mathbf{u}_1^- + \gamma_2\mathbf{u}_2^-$ and $\gamma_1\mathbf{u}_1^+ + \gamma_2\mathbf{u}_2^+$ represent equilateral triangles of any area and opposite orientations (or colocated points if $\gamma_1 = \gamma_2 = 0$), whilst $\gamma_3\mathbf{u}_3^- + \gamma_4\mathbf{u}_4^-$ and $\gamma_3\mathbf{u}_3^+ + \gamma_4\mathbf{u}_4^+$ represent colocated points, hence act as a translation for the vertices of the previous linear combination. This shows that the system is solvable if and only if $\tilde{\mathbf{v}}'$ represents an equilateral triangle of orientation $\mathrm{sign}(A^*(\tilde{\mathbf{v}}')) = s\,\mathrm{sign}(\lambda)$ or colocated vertices.

The system $X\mathbf{v}' = \tilde{\mathbf{v}}'$ is solved by first finding the solutions of the homogeneous system $X\mathbf{v}_h = 0$ and translating them by a particular solution $\mathbf{v}_p$, obtaining $\mathbf{v}' = \mathbf{v}_h + \mathbf{v}_p$. The homogeneous system has an infinite number of solutions which come from the null space of X. This can be represented as a two-dimensional linear subspace $\mathbf{v}_h = \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2$ where the coefficients $\beta_1, \beta_2 \in \mathbb{R}$ and with bases $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^6$ such that:

$$\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{2s\,\mathrm{sign}(\lambda)}{\lambda_o} \\ -\frac{2s\,\mathrm{sign}(\lambda)}{\lambda_o} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{2s\,\mathrm{sign}(\lambda)}{\lambda_o} \\ \frac{2s\,\mathrm{sign}(\lambda)}{\lambda_o} & -\frac{1}{2} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{42}$$

We have that $\mathbf{v}_1, \mathbf{v}_2$ represent centred equilateral triangles of the same area of $\frac{3}{\lambda_o}$. The linear combination $\mathbf{v}_h = \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2$ generates centred equilateral triangles of any area of orientation $-s\,\mathrm{sign}(\lambda)$. We then calculate the particular solution $\mathbf{v}_p$ using the pseudo-inverse as:

$$\mathbf{v}_p = X^{\dagger}\tilde{\mathbf{v}}' = \begin{bmatrix} \frac{\tilde{x}'_a}{2} + \frac{\tilde{x}'_b}{4} + \frac{\tilde{x}'_c}{4} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{y}'_b - \tilde{y}'_c}{3\lambda_o} \\ \frac{\tilde{y}'_a}{2} + \frac{\tilde{y}'_b}{4} + \frac{\tilde{y}'_c}{4} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{x}'_b - \tilde{x}'_c}{3\lambda_o} \\ \frac{\tilde{x}'_a}{4} + \frac{\tilde{x}'_b}{2} + \frac{\tilde{x}'_c}{4} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{y}'_a - \tilde{y}'_c}{3\lambda_o} \\ \frac{\tilde{y}'_a}{4} + \frac{\tilde{y}'_b}{2} + \frac{\tilde{y}'_c}{4} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{x}'_a - \tilde{x}'_c}{3\lambda_o} \\ \frac{\tilde{x}'_a}{4} + \frac{\tilde{x}'_b}{4} + \frac{\tilde{x}'_c}{2} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{y}'_a - \tilde{y}'_b}{3\lambda_o} \\ \frac{\tilde{y}'_a}{4} + \frac{\tilde{y}'_b}{4} + \frac{\tilde{y}'_c}{2} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\frac{\tilde{x}'_a - \tilde{x}'_b}{3\lambda_o} \end{bmatrix}. \tag{43}$$

We then translate the null space with the particular solution and obtain $\mathbf{v}' = \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2 + \mathbf{v}_p$. This linear combination generates centred triangles of any area. A noticeable property is stated in the following Lemma, whose proof is given in Appendix A.

Lemma 6. $\mathbf{v}_p = 0 \Rightarrow \mathbf{v}'$ *is an equilateral triangle. The converse is not true.*

The next step is to constrain these triangles to the prescribed area and orientation. After some minor algebraic manipulations, we obtain the signed area of the subspace as:

$$A^*(\mathbf{v}') = (1 + s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\,\mathrm{sign}(\lambda))\,\frac{A^*(\tilde{\mathbf{v}}')}{8} - s\,\mathrm{sign}(\lambda)\frac{3(\beta_1^2 + \beta_2^2)}{\lambda_o}. \tag{44}$$

When $A^*(\tilde{\mathbf{v}}') \neq 0$ we have $s\,\mathrm{sign}(\lambda) = \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))$ thus $\mathrm{sign}(\lambda) = s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))$. However, when $A^*(\tilde{\mathbf{v}}') = 0$ we have $\mathrm{sign}(A^*(\mathbf{v}')) = -s\,\mathrm{sign}(\lambda)$, which implies that $\mathrm{sign}(\lambda) = -s$. Using the orientation constraint (7), we can express $\mathbf{v}'$ as:

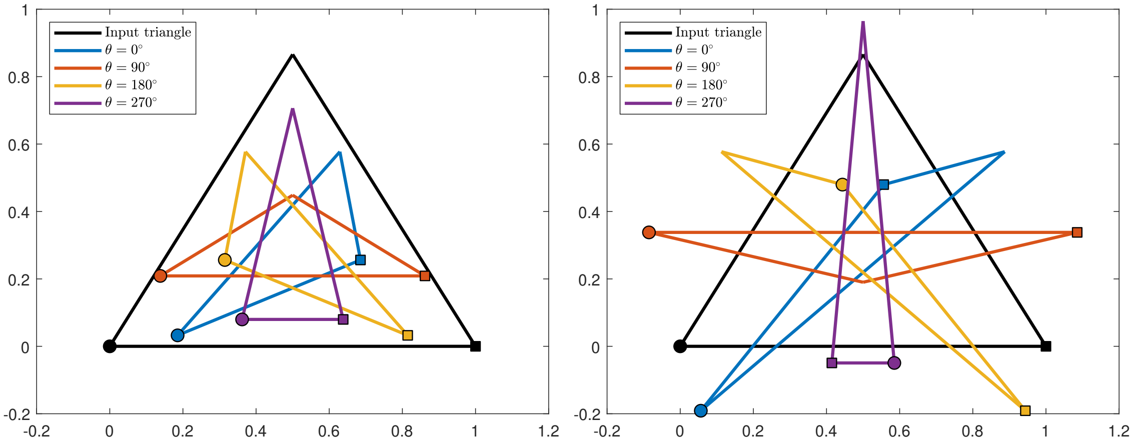
$$\begin{aligned} &\mathbf{v}' = \beta_1\mathbf{v}_1 + \beta_2\mathbf{v}_2 + \mathbf{v}_p \\ \text{s.t.}\qquad &(s + \mathrm{sign}(\lambda)\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}')))\,\frac{A^*(\tilde{\mathbf{v}}')}{8} - \mathrm{sign}(\lambda)\frac{3(\beta_1^2 + \beta_2^2)}{\lambda_o} - A_o = 0. \end{aligned} \tag{45}$$

Because of the area and orientation constraints, and because $\mathbf{v}_1$ and $\mathbf{v}_2$ are rotated copies of each other, the family defined by equation (42) can be generated by scaling $\mathbf{v}_1$ by:

$$\phi = \sqrt{\beta_1^2 + \beta_2^2} = \sqrt{\frac{\lambda_o(\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))A^*(\tilde{\mathbf{v}}') - 4kA_o)}{12}}, \tag{46}$$

where k depends on the type of input:

$$k = \begin{cases} s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}')) & \text{if } A^*(\tilde{\mathbf{v}}) \neq 0 \\ -1 & \text{if } A^*(\tilde{\mathbf{v}}) = 0, \end{cases} \tag{47}$$



(a) Orientation Preservation

(b) Orientation Inversion

Figure 2: Case II solutions for an equilateral triangle. (a), represents solutions where the input orientation is preserved and (b), where it is inverted. The prescribed area A_o is 1/5 of the input triangle's area. For both (a) and (b), four solutions corresponding to different values for θ are shown and assigned a colour for visual representation. The triangle family is generated by a free 2D rotation followed by a fixed translation in the 6D triangle space, which result in triangles that are not rotated copies of each other. Nevertheless, they all comply with the area preservation constraint and have the same cost.

so that the area constraint is met. We can then rotate $\mathbf{v}_1$ by some arbitrary angle θ. We note that when $k = s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}')) = 1$, then $\phi \in \mathbb{R}$ as long as $A(\tilde{\mathbf{v}})/4 \geq A_o$ (which corresponds to setting S_3). We define a new basis vector $\mathbf{v}_c$ as:

$$\mathbf{v}_c = \phi \begin{bmatrix} -\frac{1}{2} & -ks\frac{2}{\lambda_o} & -\frac{1}{2} & ks\frac{2}{\lambda_o} & 1 & 0 \end{bmatrix}^\top . \tag{48}$$

We translate it to the original coordinates by $\mathbf{v}_t$, which is the addition of the particular solution $\mathbf{v}_p$ and the input's centroid $\bar{\mathbf{v}}$, and obtain:

$$\mathbf{v} = \mathscr{R}(\theta)\mathbf{v}_c + \mathbf{v}_t, \tag{49}$$

where $\mathscr{R}(\theta)$ is a block diagonal matrix replicating the 2D rotation matrix $R(\theta)$ three times as $\mathscr{R}(\theta) = \mathrm{diag}(R(\theta), R(\theta), R(\theta))$. Equation (49) generates an infinite number of solutions with an identical cost, as shown by the following Lemma, proved in Appendix A.

Lemma 7. *All solutions generated by equation (49) have the same cost.*

We illustrate some solutions in Case II with a toy example in Figure 2.

2.7. Properties of the Solutions

An important property of the solutions to OTPPAO is that they preserve the centroid of the input triangle $\tilde{\mathbf{v}}$. For Case I, this is shown by substituting the vertices $\mathbf{v}$ from equation (22) in the centroid formula as:

$$\begin{aligned}\delta\bar{\mathbf{v}} = \frac{\delta}{3}\begin{bmatrix} x_a + x_b + x_c \\ y_a + y_b + y_c \end{bmatrix} &= \frac{1}{3}\left[\begin{matrix} (\lambda^2 - 16)(\tilde{x}_a + \tilde{x}_b + \tilde{x}_c) + 2\lambda^2(\tilde{x}_a + \tilde{x}_b + \tilde{x}_c) \\ (\lambda^2 - 16)(\tilde{y}_a + \tilde{y}_b + \tilde{y}_c) + 2\lambda^2(\tilde{y}_a + \tilde{y}_b + \tilde{y}_c) \end{matrix}\right. \\ &\qquad\qquad \left.\begin{matrix} +4s\lambda(\tilde{y}_b - \tilde{y}_c + \tilde{y}_c - \tilde{y}_a + \tilde{y}_a - \tilde{y}_b) \\ +4s\lambda(\tilde{x}_b - \tilde{x}_c + \tilde{x}_c - \tilde{x}_a + \tilde{x}_a - \tilde{x}_b) \end{matrix}\right] \\ &= \frac{1}{3}\begin{bmatrix} (3\lambda^2 - 16)(\tilde{x}_a + \tilde{x}_b + \tilde{x}_c) \\ (3\lambda^2 - 16)(\tilde{y}_a + \tilde{y}_b + \tilde{y}_c) \end{bmatrix} = \frac{\delta}{3}\begin{bmatrix} \tilde{x}_a + \tilde{x}_b + \tilde{x}_c \\ \tilde{y}_a + \tilde{y}_b + \tilde{y}_c \end{bmatrix}. \end{aligned} \tag{50}$$

For Case II, we similarly substitute the vertices $\mathbf{v}$ from equation (45) in the centroid formula and obtain:

$$\begin{aligned}\bar{\mathbf{v}} = \frac{1}{3}\begin{bmatrix} x_a + x_b + x_c \\ y_a + y_b + y_c \end{bmatrix} &= \begin{bmatrix} (0)\beta_1 + (0)\beta_2 s\,\mathrm{sign}(\lambda)\frac{2}{\lambda_o} \\ (0)\beta_1 s\,\mathrm{sign}(\lambda)\frac{2}{\lambda_o} + (0)\beta_2 \end{bmatrix} \\ &+ \frac{1}{3}\begin{bmatrix} \tilde{x}_a + \tilde{x}_b + \tilde{x}_c + \mathrm{sign}(\lambda)\frac{(\tilde{y}_b - \tilde{y}_c - \tilde{y}_a + \tilde{y}_c + \tilde{y}_a - \tilde{y}_b)}{\lambda_o} \\ \tilde{y}_a + \tilde{y}_b + \tilde{y}_c - \mathrm{sign}(\lambda)\frac{(\tilde{x}_b - \tilde{x}_c - \tilde{x}_a + \tilde{x}_c + \tilde{x}_a - \tilde{x}_b)}{\lambda_o} \end{bmatrix} \\ &= \frac{1}{3}\begin{bmatrix} \tilde{x}_a + \tilde{x}_b + \tilde{x}_c \\ \tilde{y}_a + \tilde{y}_b + \tilde{y}_c \end{bmatrix}. \end{aligned} \tag{51}$$

2.8. Numerical Implementation

We use the theory developed in the previous sections to construct a numerically robust procedure, given in Algorithm 1, to solve OTPPAO. In theory, the first step would be to test the input setting and then branch on Case I or Case II accordingly. However, round-off errors make the test potentially unreliable. In order to deliver a numerically robust solution, both cases must be attempted, and the optimal solution chosen a posteriori by inspecting the cost. However, an a priori case selection method for time critical but precision tolerant problems is discussed in Section 2.9. Algorithm 1 uses the input vertices $\tilde{\mathbf{v}}$, prescribed area A_o and orientation s as inputs. It also uses an area error tolerance E to handle round-off in the area constraint (8). Our method could be naturally implemented with precise arithmetic, since it provides exact solutions. However, in the context of its use within PBD, as presented in Section 3, we give it in floating-point arithmetic. Algorithm 1 starts by generating the solutions from Case I, then Case II, and chooses the optimal one. For Case I, we obtain a list $\boldsymbol{v}_1$ of at most 4 solutions. For Case II, we obtain a single best solution $\mathbf{v}_2$, the optimally rotated one, and the basis and offset to generate all solutions following equation (49). The overall optimal solution $\mathbf{v}_o$ is chosen amongst $\boldsymbol{\upsilon}_1$ and $\mathbf{v}_2$. The algorithm returns the optimal solution, along with all the solutions from Case I and Case II. This allows the user to deal with possible ambiguities and make the final choice depending on application specific priors and constraints.

Algorithm 1 Optimal Triangle Projection with a Prescribed Area and Orientation

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\mathbf{v}_o$ - optimal triangle, $\boldsymbol{\upsilon}_1$ - Case I triangle set, $\mathbf{v}_2$ - Case II optimal triangle, $\mathbf{v}_c, \mathbf{v}_t$ - Case II basis and translation
function OTTPAO($\tilde{\mathbf{v}}, A_o, s, E = 10^{-3}$)
    $\boldsymbol{\upsilon}_1 \leftarrow$ SolveCase1($\tilde{\mathbf{v}}, A_o, s, E$)                  ▷ Compute Case I solutions
    $(\mathbf{v}_2, \mathbf{v}_c, \mathbf{v}_t) \leftarrow$ SolveCase2($\tilde{\mathbf{v}}, A_o, s, E$)       ▷ Compute Case II solutions
    $\mathbf{v}_o \leftarrow$ FindTriangleOfMinimalCost($\boldsymbol{\upsilon}_1 \cup \{\mathbf{v}_2\}$)    ▷ Select the optimal solution
    return $\mathbf{v}_o, \boldsymbol{\upsilon}_1, \mathbf{v}_2, \mathbf{v}_c, \mathbf{v}_t$
end function
```

Algorithm 2 computes the possible solutions for Case I. It first computes the coefficients p, q, r of the depressed quartic equation (lines 2, 3 and 4). Then, it uses Ferrari's method, given by Algorithm 3, to find possible values of the Lagrange multiplier in vector $\boldsymbol{\lambda}$. Some values in $\boldsymbol{\lambda}$ may be complex because they do not represent a solution or because of round-off error. We thus extract the real part of $\boldsymbol{\lambda}$ (line 8). In theory, the next step would be to verify that $\lambda \neq \lambda_o$ or $\delta \neq 0$, because this would create a rank-deficiency and division by zero. However, this cannot be directly tested because of round-off error. This is better handled by taking the pseudo-inverse $\delta^\dagger = (3\lambda^2 - 16)^\dagger$ (line 10), recalling that $0^\dagger = 0$. We can then simply check that the triangle complies with the area and orientation constraints (line 11).

Algorithm 4 computes all the possible solutions for Case II. It achieves this by returning the rotational solution basis $\mathbf{v}_c$ (line 10), particular solution $\mathbf{v}_p$ (line 11) and offset $\bar{\mathbf{v}}$ (line 7). With these three components, the user can generate any solution by choosing an angle θ in equation (49). All solutions generated this way are theoretically equivalent and they all fulfil the area and orientation constraints. However, because the input vertices might not be exactly colocated numerically (which is the theoretical prerequisite of Case II for a colocated vertices input), one of the solutions in the basis may stand out as having a lower score than any other one. This solution may be, when the input vertices are close to each other, the optimal

Algorithm 2 Closed-form Analytic Solution to Case I of OTPPAO

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\upsilon_1$ - solution list
function SolveCase1($\tilde{\mathbf{v}}, A_o, s, E$)
    $p \leftarrow -\frac{16(2A_o + sA^*(\tilde{\mathbf{v}}))}{3A_o}$    ▷ Compute the coefficients of the depressed quartic
    $q \leftarrow \frac{32\sigma^2(\tilde{\mathbf{v}})}{3A_o}$
    $r \leftarrow \frac{256(A_o - sA^*(\tilde{\mathbf{v}}))}{9A_o}$
    $\boldsymbol{\lambda} \leftarrow$ FerrariSolution($p, q, r$)    ▷ Solve for the four possible Lagrange multipliers
    $\upsilon_1 \leftarrow \emptyset$    ▷ Create an empty set of solutions
    for $t \leftarrow 1, \ldots, 4$ do    ▷ Generate and select the triangles
        $\lambda \leftarrow \mathrm{Re}(\boldsymbol{\lambda}(t))$    ▷ Keep the real part
        $\delta \leftarrow 3\lambda^2 - 16$    ▷ Compute $\delta$
        $\mathbf{v} \leftarrow \delta^\dagger \begin{bmatrix} (\lambda^2-16)\tilde{x}_a + \lambda^2(\tilde{x}_b + \tilde{x}_c) + 4s\lambda(\tilde{y}_b - \tilde{y}_c) \\ (\lambda^2-16)\tilde{y}_a + \lambda^2(\tilde{y}_b + \tilde{y}_c) + 4s\lambda(\tilde{x}_c - \tilde{x}_b) \\ (\lambda^2-16)\tilde{x}_b + \lambda^2(\tilde{x}_a + \tilde{x}_c) + 4s\lambda(\tilde{y}_c - \tilde{y}_a) \\ (\lambda^2-16)\tilde{y}_b + \lambda^2(\tilde{y}_a + \tilde{y}_c) + 4s\lambda(\tilde{x}_a - \tilde{x}_c) \\ (\lambda^2-16)\tilde{x}_c + \lambda^2(\tilde{x}_a + \tilde{x}_b) + 4s\lambda(\tilde{y}_a - \tilde{y}_b) \\ (\lambda^2-16)\tilde{y}_c + \lambda^2(\tilde{y}_a + \tilde{y}_b) + 4s\lambda(\tilde{x}_b - \tilde{x}_a) \end{bmatrix}$    ▷ Compute the vertices
        if $|sA^*(\mathbf{v}) - A_o| \leq E$ then    ▷ Check the area constraint
            $\upsilon_1 \leftarrow \upsilon_1 \cup \{\mathbf{v}\}$    ▷ Add the vertices to the solution set
        end if
    end for
    return $\upsilon_1$
end function
```

Algorithm 3 Ferrari's Solution to the Depressed Quartic

```
Input: $p, q, r$ - coefficients of depressed quartic
Output: $\boldsymbol{\lambda}$ - set of four roots
function FerrariSolution($p, q, r$)
    $Q_1 \leftarrow -\frac{p^2+12r}{36}$    ▷ Compute Cardano's formula coefficients
    $Q_2 \leftarrow \frac{2p^3-72rp+27q^2}{432}$
    $\alpha_o \leftarrow \sqrt[3]{Q_2 + \sqrt{Q_1^3 + Q_2^2}} + \sqrt[3]{Q_2 - \sqrt{Q_1^3 + Q_2^2}} - \frac{p}{3}$    ▷ Compute the real root of the resolvent cubic
    $\boldsymbol{\lambda} \leftarrow \emptyset$    ▷ Create an empty solution set
    for $k_1 \leftarrow \{1, 2\}$ do
        for $k_2 \leftarrow \{1, 2\}$ do
            $\lambda \leftarrow \frac{(-1)^{k_1}\sqrt{2\alpha_o}+(-1)^{k_2}\sqrt{-\left(2p+2\alpha_o+(-1)^{k_1}\frac{2q}{\sqrt{2\alpha_o}}\right)}}{2}$    ▷ Compute the root
            $\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} \cup \{\lambda\}$    ▷ Add it to the solution set
        end for
    end for
    return $\boldsymbol{\lambda}$
end function
```

solution, even compared to Case I, owing to numerical round-off error. This solution is obtained by finding the optimal rotation for the cost function, the translation already being the optimal one, by solving:

$$\min_{R \in SO(2)} \|(\mathscr{R}\mathbf{v}_c + \mathbf{v}_t) - \tilde{\mathbf{v}}\|^2 \quad \text{with} \quad \mathscr{R} = \text{diag}(R, R, R). \tag{52}$$

This problem has a closed-form solution [18]. We first rearrange $\mathbf{v}_c$ and $\tilde{\mathbf{v}}'$ into 2×3 matrices V_c and $\tilde{V}'$. We then compute the cross-covariance matrix $W = V_c \tilde{V}'^\top$ and its SVD $W = U_1 \Sigma U_2^\top$. The optimal orthogonal matrix, which could potentially contain a reflection in addition to the rotation, is $U_2 U_1^\top$. In order to preserve the triangle orientation we restrict R to be a rotation only by setting $R = U_2 D U_1^\top$, where $D = \text{diag}(1, \det(U_1 U_2^\top))$. We finally use R to generate the optimal solution $\mathbf{v}_2$ (line 18).

2.9. A Priori Case Selection

Exact a priori case selection cannot be done to separate Cases I and II, owing to round-off errors. However, an approximate numerical approach can be implemented using a numerical tolerance. First, we design a measure which determines if a triangle

Algorithm 4 Closed-form Analytic Solution to Case II of OTPPAO

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\mathbf{v}_2$ - optimal triangle, $\mathbf{v}_c, \mathbf{v}_t$ - triangle basis and translation
function SolveCase2($\tilde{\mathbf{v}}, A_o, s, E$)
    if $|A^*(\mathbf{v})| \leq E$ then                                  ▷ Check the Input's area
        $k \leftarrow s\,\mathrm{sign}(A^*(\mathbf{v}))$                 ▷ Compute $k$ for an equilateral triangle
    else
        $k \leftarrow -1$                                               ▷ Compute $k$ for a single point
    end if
    $\bar{\mathbf{v}} \leftarrow \frac{1}{3}\begin{bmatrix} \tilde{x}_a + \tilde{x}_b + \tilde{x}_c \\ \tilde{y}_a + \tilde{y}_b + \tilde{y}_c \end{bmatrix}$    ▷ Compute the centroid of the input vertices
    $\tilde{\mathbf{v}}' \leftarrow \tilde{\mathbf{v}} - \bar{\mathbf{v}}$          ▷ Translate the input vertices
    $\phi \leftarrow \sqrt{\frac{\lambda_o(\mathrm{sign}(A^*(\tilde{\mathbf{v}}))A^*(\tilde{\mathbf{v}})-4kA_o)}{12}}$    ▷ Computes the area constraint parameter
    $\mathbf{v}_c \leftarrow \mathrm{Re}(\phi)\begin{bmatrix} -\frac{1}{2} & -ks\frac{2}{\lambda_o} & -\frac{1}{2} & ks\frac{2}{\lambda_o} & 1 & 0 \end{bmatrix}^\top$    ▷ Compute the solution basis
    $\mathbf{v}_p \leftarrow \begin{bmatrix} \frac{\tilde{x}'_a}{2} + \frac{\tilde{x}'_b}{4} + \frac{\tilde{x}'_c}{4} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{y}'_b - \tilde{y}'_c}{3\lambda_o} \\ \frac{\tilde{y}'_a}{2} + \frac{\tilde{y}'_b}{4} + \frac{\tilde{y}'_c}{4} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{x}'_b - \tilde{x}'_c}{3\lambda_o} \\ \frac{\tilde{x}'_a}{4} + \frac{\tilde{x}'_b}{2} + \frac{\tilde{x}'_c}{4} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{y}'_a - \tilde{y}'_c}{3\lambda_o} \\ \frac{\tilde{y}'_a}{4} + \frac{\tilde{y}'_b}{2} + \frac{\tilde{y}'_c}{4} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{x}'_a - \tilde{x}'_c}{3\lambda_o} \\ \frac{\tilde{x}'_a}{4} + \frac{\tilde{x}'_b}{4} + \frac{\tilde{x}'_c}{2} + \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{y}'_a - \tilde{y}'_b}{3\lambda_o} \\ \frac{\tilde{y}'_a}{4} + \frac{\tilde{y}'_b}{4} + \frac{\tilde{y}'_c}{2} - \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{\tilde{x}'_a - \tilde{x}'_b}{3\lambda_o} \end{bmatrix}$    ▷ Compute the particular solution
    $\tilde{V}' \leftarrow$ rearrange $\tilde{\mathbf{v}}'$ into a $2 \times 3$ matrix
    $V_c \leftarrow$ rearrange $\mathbf{v}_c$ into a $2 \times 3$ matrix
    $(U_1, \Sigma, U_2) \leftarrow \mathrm{SVD}\left(\tilde{V}'V_c^\top\right)$          ▷ Compute the optimal rotation
    $D \leftarrow \mathrm{diag}(1, \det(U_1 U_2))$
    $\mathbf{v}_t \leftarrow \mathbf{v}_p + \bar{\mathbf{v}}$                      ▷ Compute the translation vector
    $R \leftarrow U_2 D U_1^\top$
    $\mathbf{v}_2 \leftarrow \mathrm{diag}(R, R, R)\mathbf{v}_c + \mathbf{v}_t$     ▷ Compute the optimal solution
    return $\mathbf{v}_2, \mathbf{v}_c, \mathbf{v}_t$
end function
```

is equilateral or if its vertices are colocated as, according to Proposition 1, this forms a necessary condition for Case II. For that, we compute the edge lengths as d_{ab}, d_{ac} and d_{bc} and then the maximum of the absolute edge length differences. Introducing a numerical tolerance τ_1, this leads to the following necessary condition for Case II: $\max(|d_{ab} - d_{ac}|, |d_{ab} - d_{bc}|, |d_{ac} - d_{bc}|) \leq \tau_1$. Still following Proposition 1, we compute $A(\tilde{\mathbf{v}})$ and $s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))$ to determine if $\tilde{\mathbf{v}}$ corresponds to any of the three settings of Case II. The detailed implementation of a priori case selection is given in Algorithm 5.

The numerical implementation critically depends on the two tolerance values, τ_1 and τ_2. There is thus a risk that this test does not select the optimal solution for all inputs, in particular inputs being very close to an equilateral triangle, but not quite equilateral, and for which the optimal solution would be given by Case I and not Case II. Consequently, any application relying on this method must tune τ_1 and τ_2 to maximise the instances of true positive (Case II is selected when $\mathscr{C}(\boldsymbol{v}_1) \geq \mathscr{C}(\mathbf{v}_2)$)) whilst minimising the instances of false negative selection (Case I is selected when $\mathscr{C}(\boldsymbol{v}_1) \geq \mathscr{C}(\mathbf{v}_2)$)). Furthermore, if τ_1 is very small and Case I is wrongfully selected, then we would have $\lambda \approx \lambda_o$ and the solution would not comply with the constraint $sA^*(\tilde{\mathbf{v}}) - A_o \leq E$, causing significant error.

Algorithm 5 A priori case selection

Input: $\tilde{\mathbf{v}}$ - input vertices, A_o - prescribed area, s - prescribed orientation, τ_1 - case separator tolerance 1, τ_2 - case separator tolerance 2

Output: selected_case - case selection variable

```
function CaseSelection($\tilde{\mathbf{v}}, A_o, s, \tau, E$)
    $d_{ab} \leftarrow |v_a - v_b|$                                        ▷ Edge lengths
    $d_{ac} \leftarrow |v_a - v_c|$
    $d_{bc} \leftarrow |v_b - v_c|$
    if $\max(|d_{ab} - d_{ac}|, |d_{ac} - d_{bc}|, |d_{ab} - d_{bc}|) < \tau_1$ then      ▷ Case Selection
        if $A(\tilde{\mathbf{v}}) < \tau_2$ or $s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}})) = -1$ or $A(\tilde{\mathbf{v}})/4 \geq A_o$ then
            selected_case ← 2
        else
            selected_case ← 1
        end if
    else
        selected_case ← 1
    end if
    return selected_case
end function
```

2.10. Numerical Examples

We show the results of our algebraic procedure in a series of illustrative examples presented in Tables 2 and 3. Each row represents an example with a different type of input. The first column contains the input parameters (input vertices $\tilde{\mathbf{v}}$ with area $A^*(\tilde{\mathbf{v}})$, prescribed area A_o and orientation s). The second column shows the cost $\mathscr{C}(\mathbf{v})$ and the generated area $A^*(\mathbf{v})$. For these examples, we opted to show the four solutions from Case I and the optimally rotated solution from Case II, and highlight the overall optimal solution. The third column shows the input triangle and the generated solution triangles. We draw a circle and a square in two of the vertices of the triangles to visualise the potential inversions.

In Table 2, the inputs are random general triangles. For each example we want to find the optimal triangle that has the prescribed area and orientation. The first and second examples are triangles whose orientation matches the prescribed orientation, meaning that $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$, while the third example represents the opposite case, meaning that $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = -s$. The inputs in the second and third examples are identical, except for the prescribed orientation s. In all three examples, for Case I, we observe that the first and second solutions respect the signed area constraint while the third and fourth do not. This happens because the third and fourth roots of the depressed quartic are complex and the vertices produced by these solutions are altered once we extract their real part in Algorithm 2. We also observe that the third and fourth solutions of Case I are the same. The reason is that they correspond to complex conjugate roots, and thus have the same real part. The solutions given by Case II also respect the area constraint. In the first and second examples, the vertices of the second solution of Case I and the solution of Case II are close to the input vertices, resulting in lower costs, however the solution of Case II is always an equilateral triangle. In both examples, the minimal cost is given by the second solution of Case I and is considered optimal. In the third example, the resulting vertices are not simple inversions of the previous solutions but new solutions that are accommodated to the prescribed orientation. In this case an optimal solution is also found, albeit at a higher cost.

In Table 3, the inputs are special configurations. The first example represents a flat triangle with colinear input vertices. In this instance, our algorithm behaves as expected, similarly to the examples with non-flat triangles, and returns an optimal solution (solutions 3 and 4 of Case I return a triangle considerably larger than the input triangle). The second example represents an example where all the input vertices are colocated ($\tilde{v}_a = \tilde{v}_b = \tilde{v}_c$). In this instance, Case I solutions are ignored and the optimal solution is given by Case II. The solution given by Case II can be rotated at any angle but the cost remains constant. The third example is for an equilateral triangle whose orientation is the opposite of the prescribed orientation. In this example, none of the solutions given by Case I respects the area constraint and δ is very small (especially in solutions 3 and 4 of Case I where $\delta^{\dagger}$ is close to zero and thus returns a triangle 10^{15} times larger than the input triangle). The optimal solution is given by Case II and $\mathbf{v}_c$ can be rotated at any angle, producing different triangles with the cost remaining constant. In the end, our algorithmic procedure always computes the optimal solution for all six examples.

3. Area-based 2D Mesh Editing

We use our method in triangular 2D mesh editing. The implementation is similar to PBD [6] but instead of linearising the area constraint, we perform an optimal projection for each triangle in the mesh. The N_p mesh vertices are in $\mathbf{P} \in \mathbb{R}^{N_p \times 2}$ and the N_t triangles in $\mathbf{M} \in \mathbb{R}^{N_t \times 3}$ with prescribed areas $\mathbf{A}_o \in \mathbb{R}^{N_t}$ and prescribed orientation $\text{sign}(A^*(\tilde{\mathbf{v}}))$. The implementation is given in Algorithm 6.

Algorithm 6 Prescribed Area Preservation 2D Mesh PBD

```
Input: $\mathbf{P}$ - mesh vertices, $\mathbf{M}$ - triangle indices, $\mathbf{A}_o$ - prescribed areas, $T_c$ - displacement threshold, $E$ - area error tolerance
Output: $\mathbf{P}$ - edited mesh points
$C \leftarrow \infty$
while $C \geq T_c$ do                                          ▷ Iterate until convergence
    $\tilde{\mathbf{P}} \leftarrow \mathbf{P}$                                          ▷ Copy the mesh vertices
    for $t \leftarrow 1, \dots, N_t$ do
        $p \leftarrow \mathbf{M}(t,:)$                                     ▷ Indices of the triangle
        $\tilde{\mathbf{v}} \leftarrow \mathbf{P}(p,:)$                                  ▷ Coordinates of the triangle
        $A_o \leftarrow \mathbf{A}_o(t)$                                  ▷ Prescribed area of the triangle
        $s \leftarrow \text{sign}(A^*(\tilde{\mathbf{v}}))$                          ▷ Orientation of the triangle
        $\mathbf{v}_o \leftarrow \text{OTTPAO}(\tilde{\mathbf{v}}, A_o, s, E)$                   ▷ Optimal projection
        $\mathbf{P}(p,:) \leftarrow \mathbf{v}_o$                                 ▷ Update mesh points
    end for
    $C \leftarrow \frac{1}{N_t} \sum_{i=1}^{N_t} \|\tilde{\mathbf{P}}(i,:) - \mathbf{P}(i,:)\|$            ▷ Average displacement
end while
```

3.1. Shape Dataset

For our dataset, we used Distmesh [19] to create a set of synthetic triangular meshes. As shown in Figure 3, the shapes ranged from simple convex shapes to

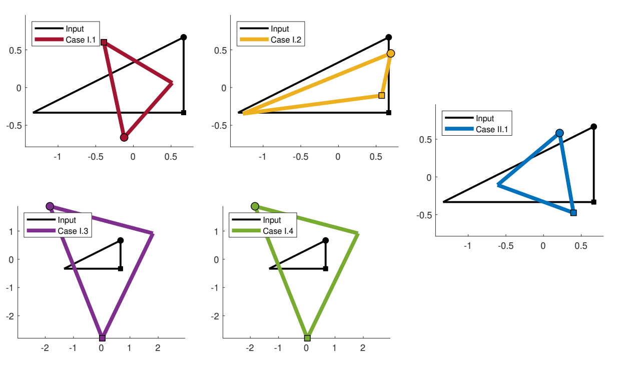
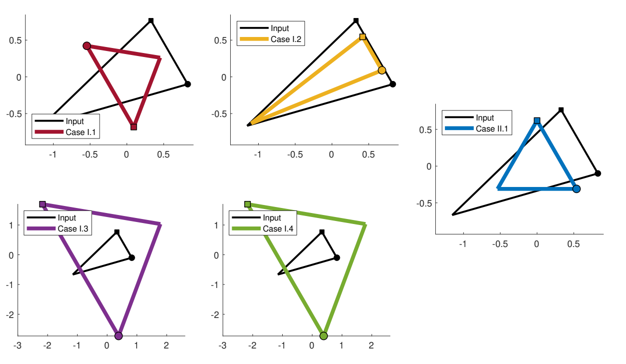
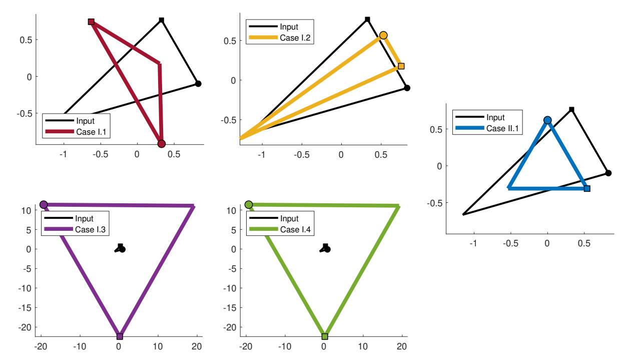
Input	Output			Generated Triangles
Input: Negative Oriented Triangle	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = -1$	2.820	-0.500	🟥	
	0.334	**-0.500**	🟨	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.666 & 0.666 \\ 0.666 & -0.333 \\ -1.333 & -0.333 \end{bmatrix}$	5.065	7.629	🟪	
	5.065	7.629	🟩	
$A^*(\tilde{\mathbf{v}}) = -1.000$	Case II			
$A_o = 0.500$	0.937	-0.500	🟦	
Input: Positive Oriented Triangle	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = 1$	2.785	0.500	🟥	
	0.345	**0.500**	🟨	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.827 & -0.100 \\ 0.327 & 0.766 \\ -1.155 & -0.667 \end{bmatrix}$	5.078	-7.919	🟪	
	5.078	-7.919	🟩	
$A^*(\tilde{\mathbf{v}}) = 1.000$	Case II			
$A_o = 0.500$	0.875	0.500	🟦	
Input: Positive Oriented Triangle	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = -1$	2.158	-0.500	🟥	
	1.041	**-0.500**	🟨	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.827 & -0.100 \\ 0.327 & 0.766 \\ -1.155 & -0.667 \end{bmatrix}$	40.35	648.58	🟪	
	40.35	648.58	🟩	
$A^*(\tilde{\mathbf{v}}) = 1.000$	Case II			
$A_o = 0.500$	1.707	-0.500	🟦	

Table 2: Numerical examples with single triangles. The left column shows the type of input, prescribed orientation s, input vertices $\tilde{\mathbf{v}}$, input area $A^*(\tilde{\mathbf{v}})$ and prescribed area A_o. The middle column shows the cost and area obtained for the triangles computed by our algebraic procedure. All four solutions from Case I and one solution from Case II are shown, assigned a colour for visual representation and the optimal solution is highlighted. The right column shows the computed triangles superimposed with the input triangle.

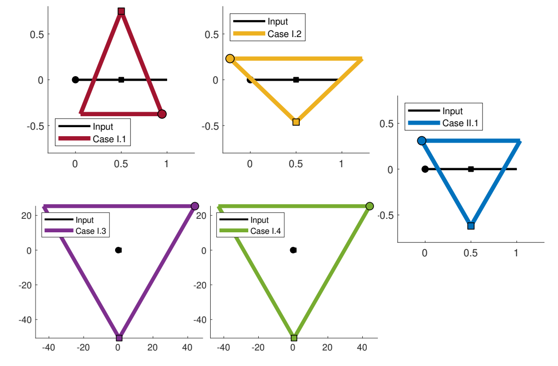
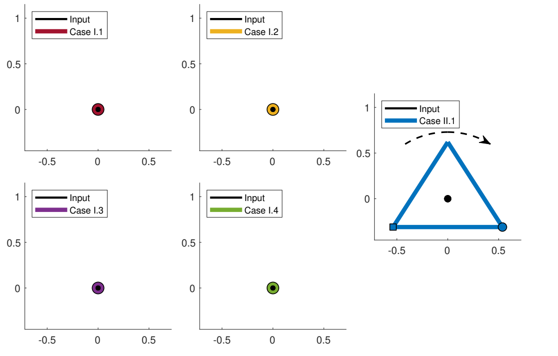
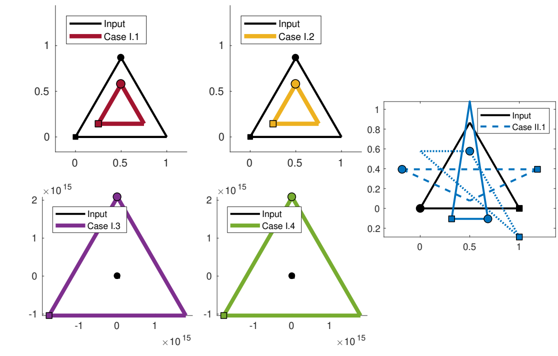
Input	Output			Generated Triangles
Input: Colinear Vertices	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = 1$	1.621	0.500	🟥	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.000 & 0.000 \\ 0.500 & 0.000 \\ 1.000 & 0.000 \end{bmatrix}$	0.647	0.500	🟨	
	88.049	-3318	🟪	
	88.049	-3318	🟩	
$A^*(\tilde{\mathbf{v}}) = 0.000$	Case II			
$A_o = 0.500$	0.762	0.500	🟦	
Input: Colocated Vertices	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = 1$	0.000	0.000	🟥	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.000 & 0.000 \\ 0.000 & 0.000 \\ 0.000 & 0.000 \end{bmatrix}$	0.000	0.000	🟨	
	0.000	0.000	🟪	
	0.000	0.000	🟩	
$A^*(\tilde{\mathbf{v}}) = 0.000$	Case II			
$A_o = 0.500$	1.075	0.500	🟦	
Input: Positive Oriented Equilateral Triangle	Cost	$A^*(\mathbf{v})$	⬜	
	Case I			
$s = -1$	0.500	0.108	🟥	
$\tilde{\mathbf{v}} = \begin{bmatrix} 0.000 & 0.000 \\ 1.000 & 0.000 \\ 0.500 & 0.866 \end{bmatrix}$	0.500	0.108	🟨	
	$3.60\cdot10^{15}$	$5.62\cdot10^{30}$	🟪	
	$3.60\cdot10^{15}$	$5.62\cdot10^{30}$	🟩	
$A^*(\tilde{\mathbf{v}}) = 0.433$	Case II			
$A_o = 0.216$	1.000	-0.216	🟦	

Table 3: Numerical examples with special triangles. The left column shows the type of input, prescribed orientation s, initial vertices $\tilde{\mathbf{v}}$, input area $A^*(\tilde{\mathbf{v}})$ and prescribed area A_o. The middle column shows the cost and area obtained for the triangles computed by our algebraic procedure. All four solutions from Case I and one solution from Case II are shown and assigned a colour for visual representation. The right column shows the computed triangles superimposed with the input triangle. In the case of colocated or equilateral input vertices, the generated equilateral triangle can be rotated arbitrarily without changing the cost.

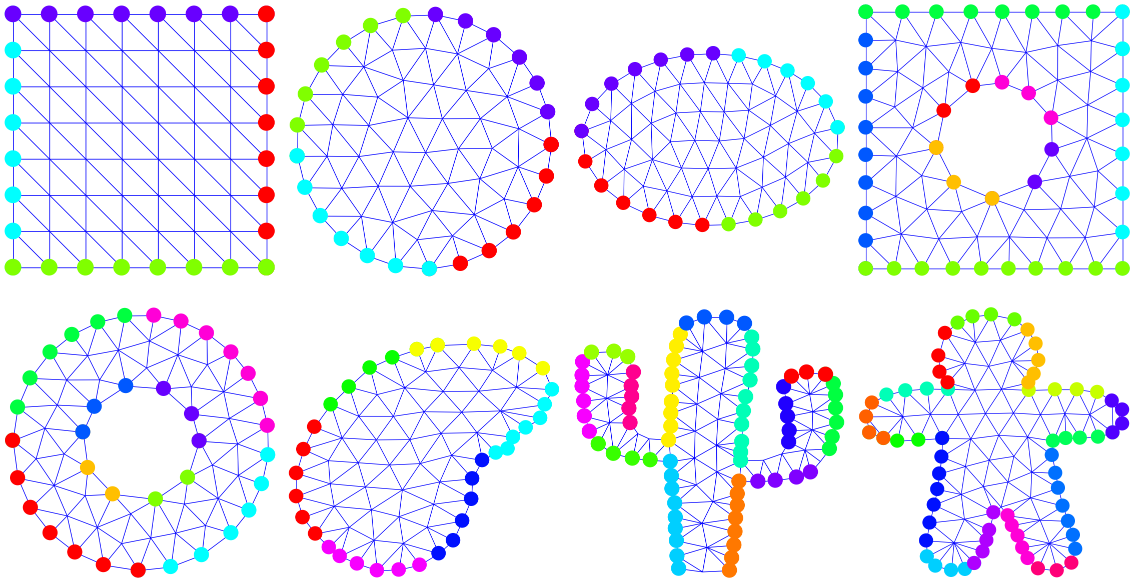


Figure 3: Shapes from the coarse subset of our synthetic polygonal mesh dataset. The vertices located at the edge are divided in sets of connected vertices represented with the same colour.

nonconvex shapes with different levels of complexity. The dataset was divided into two subsets: one subset of 8 coarse meshes composed approximately of 100 triangles and one subset of 8 fine meshes composed approximately 1000 triangles. The meshes were designed so the distances between connected vertices were approximately the same. Our method finds the optimal triangle projection and is as such independent of the input triangle size. Consequently, meshes with different triangle sizes are seamlessly handled. We use the areas of each of the triangles as prescribed areas A_o.

3.2. Generating Deformation Constraints

For our experiments we require the magnitude of the initial deformation. Since we deal with nonconvex shapes of different levels of complexity, size and orientation, we normalise this magnitude with respect to the maximum distance between two vertices in the direction of maximum variance. We treat the meshes as 2D point clouds and calculate the 95% confidence ellipse that surrounds the vertices [20]. We take the maximum distance D as twice the length of the semi-major axis of the ellipse. We divide the vertices located at the edge of the polygonal mesh in sets of connected vertices that represent a line or curve, as shown in Figure 3. Then, we apply an initial deformation by translating a given set of edge vertices in a random direction that does not cause self-collision. The magnitude of this translation is a fraction of D.

3.3. Methodology

We test the effectiveness of PBD-opt by applying an initial deformation to a synthetic mesh and measuring the number iterations it takes to converge compared to PBD-lin on the exact same generated data. Convergence is achieved when the average displacement of the mesh vertices is lower than some displacement threshold T_c. We test both methods with the coarse and fine mesh datasets. For each dataset, we perform 200 random deformations per mesh (1600 deformations in total). We applied initial deformations to the meshes of 5%, 10% and 20% of the maximum inter-vertex distance D. We measure the convergence speed as the number of iterations it takes to reach three different displacement thresholds T_c at 5%, 2.5% and 1% of D. A run stops when a method's cost reaches the lowest threshold ($T_c = 1\%$) or after it reaches a stopping time (10^4 iterations[3]).

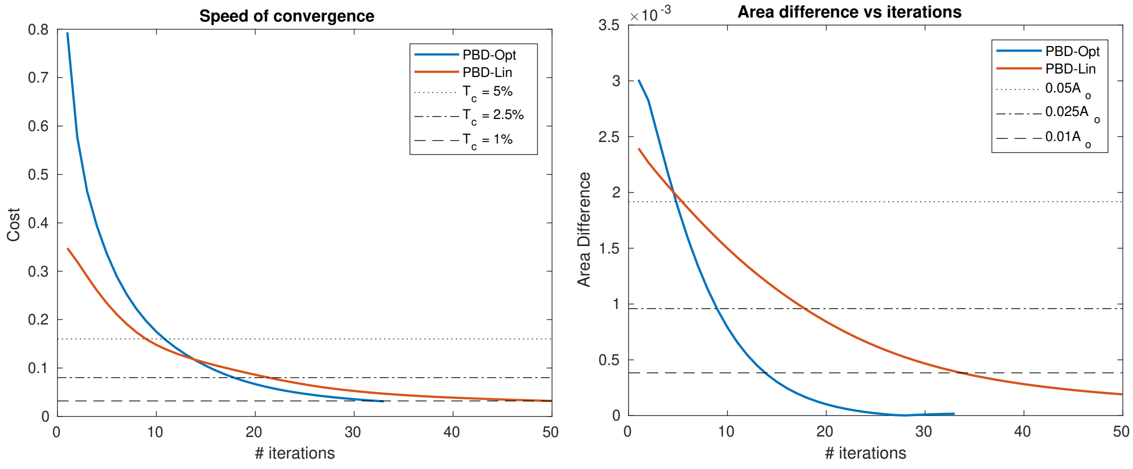
We illustrate our methodology with an example presented in Figure 4. The input is a circle shaped coarse mesh with an initial random deformation of 10% of maximum inter-vertex distance D. As can be seen in Figure 4a, the initial displacement cost of PBD-lin is lower than PBD-opt, however after some iterations the cost of PBD-opt becomes lower while the cost of PBD-lin takes many more iterations to converge. In Figure 4b, we show the evolution of the triangle area preservation constraint by comparing the difference of between the mesh triangle areas and prescribed areas. We observe that, by the time PBD-lin reaches convergence, its area difference is larger compared to PBD-opt.

3.4. Results

Results can be found in Figures 5 and 6. In the left column of each Figure we use box plots to compare the median and variability of convergence speed of both methods according to their deformation and displacement threshold. Due to the large number of outliers obtained, especially with PBD-lin, we decided not to include them in the box plots but rather to represent them in stacked bar graphs in the right column of each Figure.

For the coarse database we observe that the median convergence time for PBD-opt is higher compared to PBD-lin for a threshold of 5% and relatively similar or lower for 2.5% and 1%. However, since all the box plot pairs overlap each other, we

[3] The stopping time choice was arbitrary. However considering that the convergence speed of PBD-opt was lower than 1000 iterations, the stopping time is sufficiently high for our experiments.



(a) Evolution of cost

(b) Evolution of area difference

Figure 4: Displacement cost and area difference comparison of mesh-editing for both PBD-lin (red) and PBD-opt (blue) across the iterations. (a), displacement thresholds are used to quantify the evolution of convergence speed (dotted and dashed black lines). (b), area thresholds are used to quantify the evolution of the constraint convergence speed.

cannot conclude with 95% confidence that the medians differ. On the other hand, we observe that the results of PBD-opt are more stable since they have either similar or smaller variances compared to the results of PBD-lin. These differences are further exacerbated when evaluating the fine meshes dataset where the variance of PBD-lin is many times higher compared to the variance in PBD-opt.

For the outlier analysis we must make a distinction between two types of outliers. The first type are Slow Convergence (SC) outliers, which surpasses the upper limit of the box plot but did not reach the stopping time. The second type are Very Slow Convergence (VSC) outliers, which reach the stopping time. For PBD-opt, at most 2% of the runs were SC outliers and 0% were VSC outliers. However for PBD-lin, in the coarse meshes dataset, 10% of the runs were SC outliers. In the fine meshes dataset, around 34% of the runs were VSC outliers. This means that PBD-lin has a higher risk of getting stuck in iterations whose convergence time would be way higher compared to their median convergence speed. PBD-opt, on the other hand, provides more stable results with significantly fewer outliers.

3.5. Timing Evaluation

We experimentally compare the computation time of our method PBD-opt and the PBD-lin baseline. Whilst our original implementation used Matlab, the timing was done with a C implementation, required for realistic timing assessment. Then,

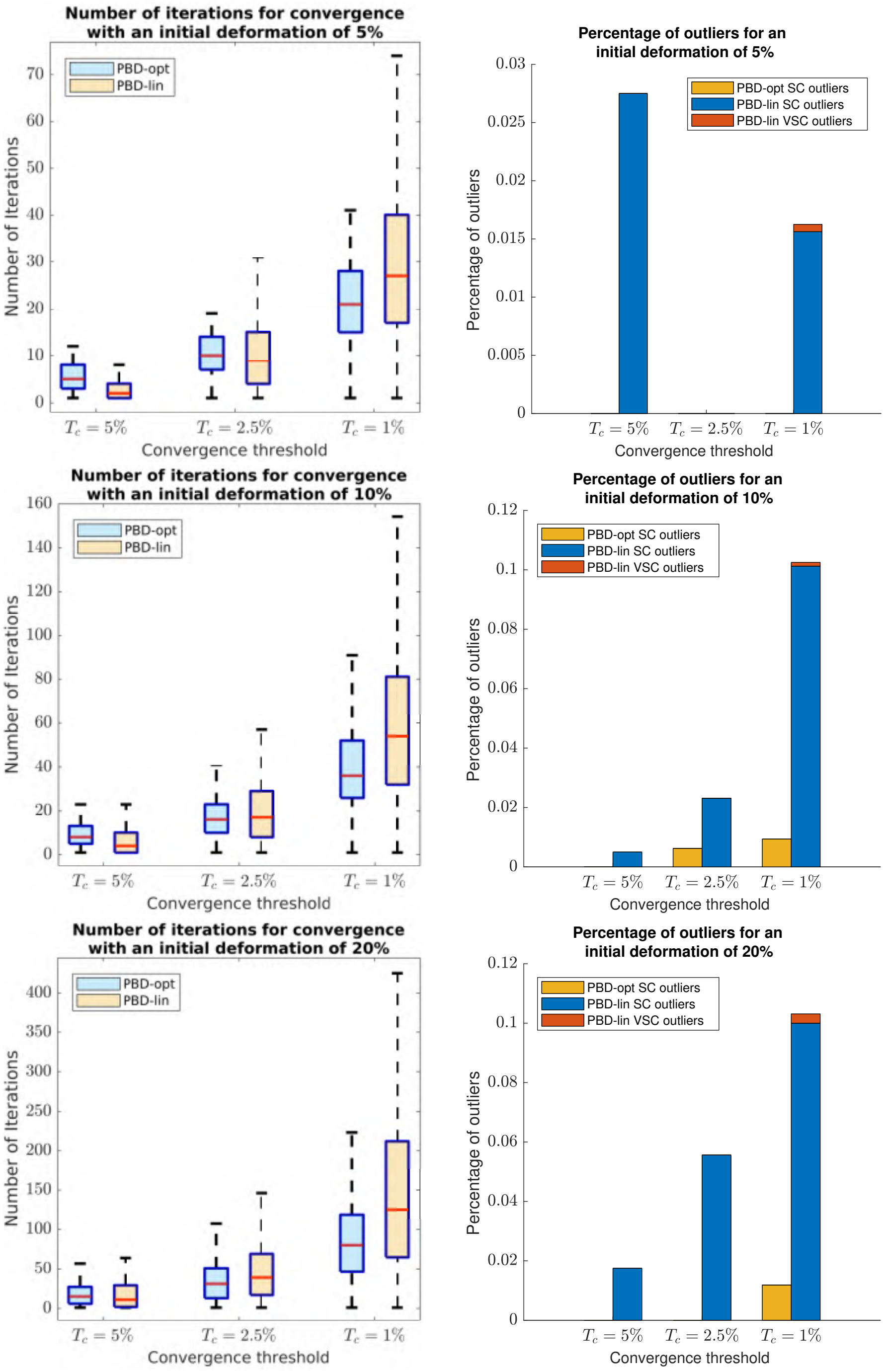


Figure 5: Convergence speed results for coarse 2D meshes. The left column shows the statistics for the number of iterations to reach conversion (omitting outliers). The right column shows the proportion of Slow Convergence (SC) and Very Slow Convergence (VSC) outliers per method.

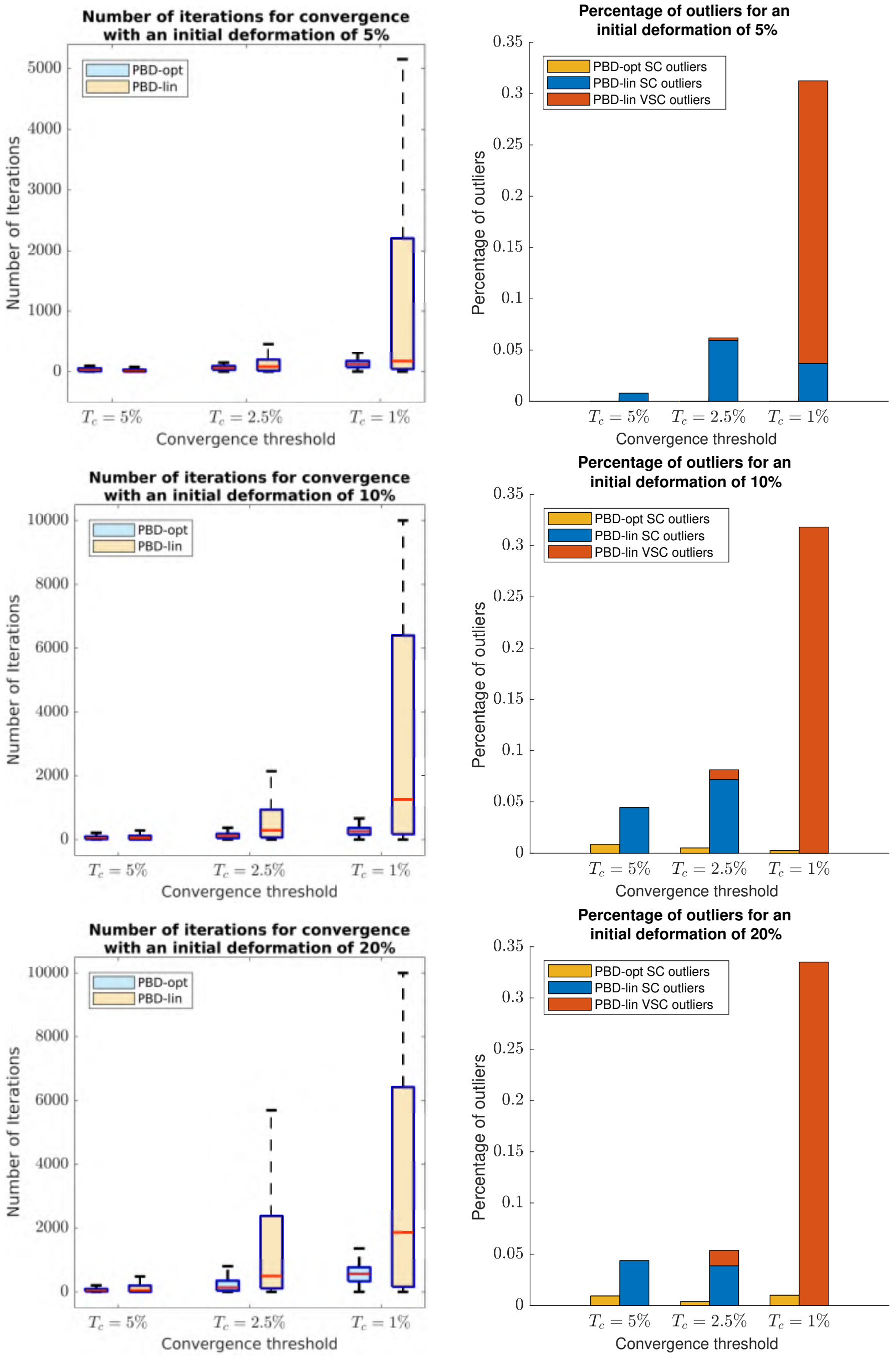


Figure 6: Convergence speed results for fine 2D meshes. The left column shows the statistics for the number of iterations to reach conversion (omitting outliers). The right column shows the proportion of Slow Convergence (SC) and Very Slow Convergence (VSC) outliers per method.

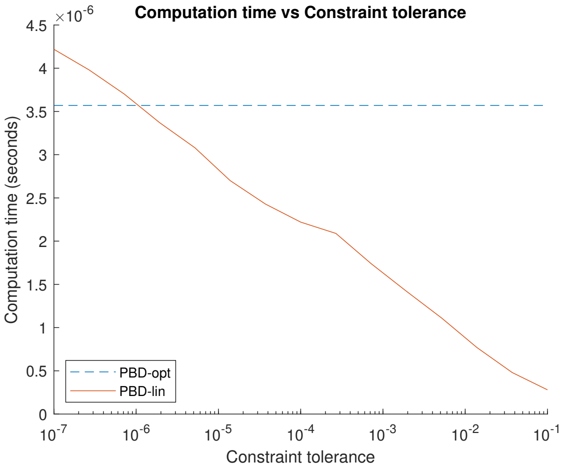


Figure 7: Computation time comparison.

we simulated 10^5 random triangles with normalised vertices $\tilde{\mathbf{v}} \in [0, 1]$ and measured the computation time. For PBD-opt the computation time was found to be $3.57 \cdot 10^{-6}$ seconds. The computation time for one iteration of PBD-lin is $3.79 \cdot 10^{-7}$ seconds, which is an order of magnitude faster than PBD-opt. However, the total computation time for PBD-lin depends on the number of iterations required to converge, which depends on the area constraint fulfilment, hence on the constraint tolerance. Because different applications have different expectations in terms of numerical constraint fulfilment, we ran PBD-lin with a varying constraint tolerance in the interval $[10^{-7}, 10^{-2}]$ and measured computation time for each tolerance value. The results are presented in Figure 7. As can be seen, the computation time of PBD-lin is inversely related to the constraint tolerance, while the computation time of PBD-opt remains fixed, as expected. For low values of the constraint tolerance, which are cases where we want to be strict on the area constraint in the result, PBD-lin takes longer to converge than the optimal PBD-opt. However, when slack is introduced by increasing the constraint tolerance, which are cases where we tolerate some inaccuracy in the result of PBD-lin, PBD-lin runs faster than PBD-opt. Note that in some instances, PBD-line did not manage to fulfill the constraint (which are the VSC outliers from Section 3.4) and we omitted these instances from the graph. In short, we can say that, in order to achieve a result on par with PBD-opt in terms of precision, PBD-lin takes a longer runtime in the cases where it converges properly. However, whilst PBD-opt has a fixed computation budget, PBD-lin can trade-off runtime and accuracy.

3.6. Use-cases

We present a series of use-cases where the triangle area and orientation constraints are combined with additional elements such as forces and other constraints that may be used in mesh editing. The selected elements are pin constraints (some selected vertices are fixed during the deformation), edge length preservation and gravity. For each use-case, we apply an initial deformation to a synthetic mesh, apply PBD-opt and the additional forces and constraints until it converges, and compare it to PBD-lin on the same generated data. Similarly to Section 3.3, convergence is achieved when $T_c = 1\%$. The notation for the different use-cases is shown in Figure 8. Note that the initial displacement and pin constraints are applied to specific vertices, whilst area, orientation, distance preservation constraints and gravity are applied to all the vertices in the mesh. For each use-case we present the initial state, the resulting deformation using both methods, the evolution of the convergence speed, area constraint satisfaction and total displacement as shown in Figures 9, 10 and 11. Our selected use-cases are as follows:

- **Use-case 1.** In this use-case, area and orientation must be preserved and some pin constraints are applied. The results are shown in Figure 9. As can be seen, PBD-opt converges faster than PBD-lin, while maintaining overall a lower constraint satisfaction error. We also observe that under PBD-opt the mesh vertices have an initial fast displacement until they reach a plateau, while under PBD-lin they keep moving in further iterations. In this use-case, we observe that some triangles of PBD-lin reach a local minimum, as evidenced by the unexpected deformation seen in the lower right side of the mesh, which makes the overall process converge slower and have a higher constraint satisfaction error.

- **Use-case 2.** In this use-case, area, orientation and edge length must be preserved. The results are shown in Figure 10. As can be seen, the overall deformation, speed of convergence, constraint satisfaction and total displacement are almost the same for PBD-opt and PBD-lin. This is explained by the edge length preservation constraint, which has a strong impact on the deformation, making the area constraint less significant and resulting in very rigid deformations.

- **Use-case 3.** In this use-case, area and orientation must be preserved, some pin constraints are applied and gravity is added. Gravity is implemented as a small displacement added to every non-pinned vertices of magnitude $9.8{\cdot}10^{-4}$ at every iteration. Due to this constant displacement, it is unlikely that the simulation

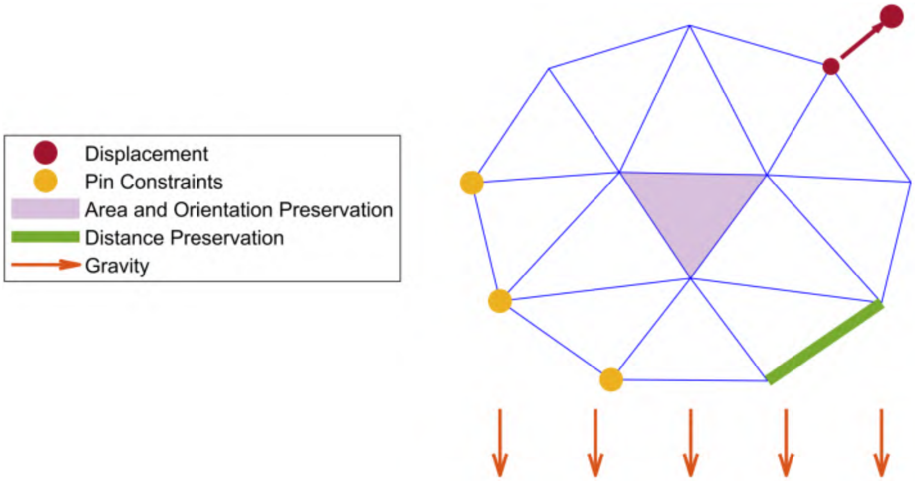


Figure 8: Use-case initial deformation and constraint notation.

reaches convergence at $T_c = 1\%$, and we thus stopped the simulation after 300 iterations. For both PBD-opt and PBD-lin, the cost and total displacement are very similar, however, PBD-opt has an overall lower constraint satisfaction error. The convergence profile for PBD-opt decreases consistently, while PBD-lin increases after iteration 200. Lastly, the simulation result is much more visually pleasing for PBD-opt than PBD-lin.

4. Conclusion

We have identified two problems related to finding the closest triangle to an input triangle under a prescribed area constraint (the OTPPA problem) and under a prescribed area and orientation constraints (the OTPPAO problem). We have given a detailed analysis and a closed-form solution to both of these problems for the first time. We have then developed a numerically robust algebraic implementation. We have used it within Point-Based Dynamics, resulting in a 2D triangular mesh editing procedure which has been shown to be faster and more stable than the existing method.

Our method forms a basis to solve further related problems in the 3D space. For a triangle in the 3D space, we can trivially apply a rigid transformation to take this triangle to one of the basis planes, calculate the optimal projection with our proposed 2D method and transform the vertices back to the original plane. A more complex extension would be for a tetrahedron in the 3D space and the constraint of a prescribed volume. This problem can be formulated similarly to OTTPAO. However,

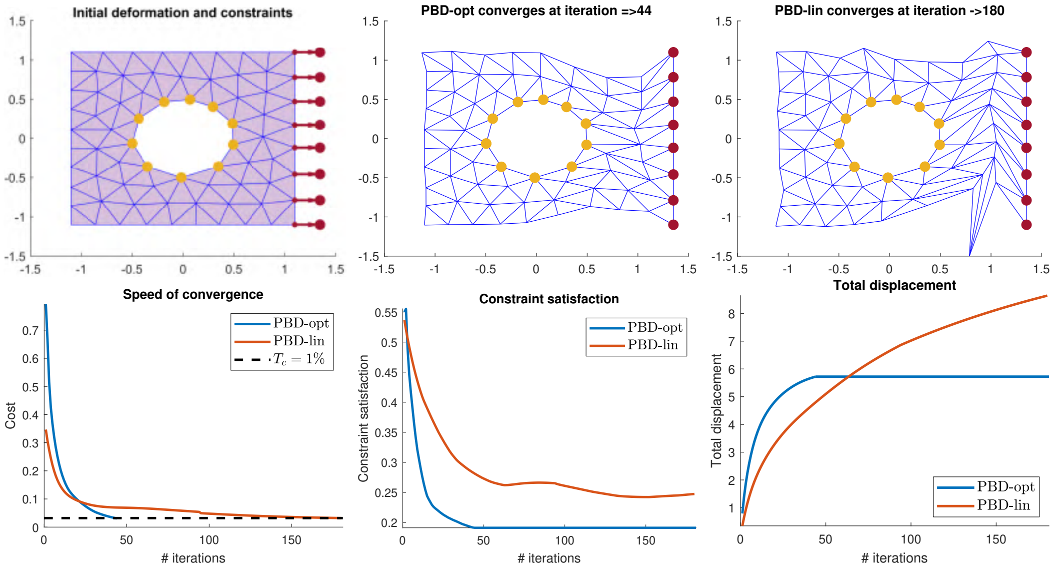


Figure 9: Results for use-case 1, with area and orientation preservation and pin constraints. The first row shows (from left to right) the initial deformation and constraints, the results from PBD-opt and PBD-lin. The second row shows the evolution of convergence speed, area constraint satisfaction and total displacement.

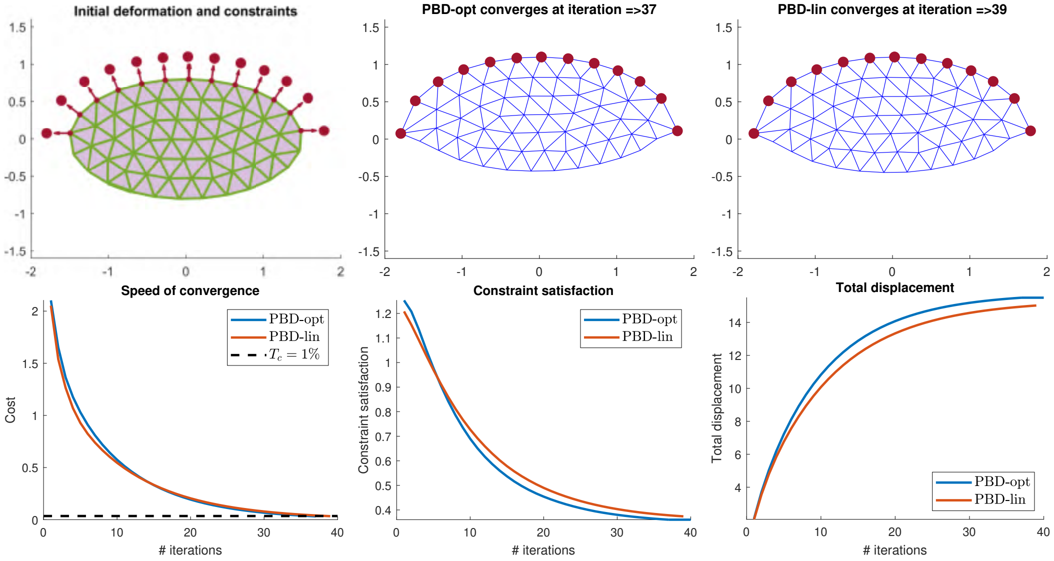


Figure 10: Results for use-case 2, with area, orientation and distance preservation. The first row shows (from left to right) the initial deformation and constraints, the results from PBD-opt and PBD-lin. The second row shows the evolution of convergence speed, area constraint satisfaction and total displacement.

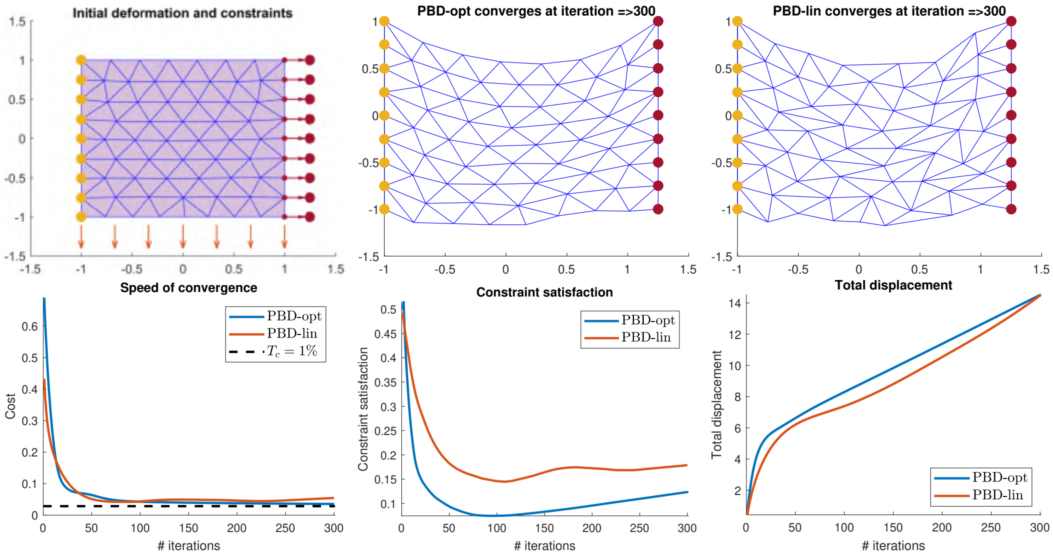


Figure 11: Results for use-case 3, with area and orientation preservation, pin constraints and gravity. The first row shows (from left to right) the initial deformation and constraints, the results from PBD-opt and PBD-lin. The second row shows the evolution of convergence speed, area constraint satisfaction and total displacement.

it differs by the constraint it involves. While the area constraint in OTTPAO leads to a quadratic equation, the volume constraint leads to a cubic equation. Therefore, while OTTPAO provides an initial step towards solving this problem, the extension is not a trivial one and forms the subject of future work.

Appendices

Appendix A. Proof of Lemmas

Proof of Lemma 1. We start with the forward implication: $S_1 \Rightarrow A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = \lambda_o$. In S_1, $\tilde{\mathbf{v}}$ represents a single point. This implies $A(\tilde{\mathbf{v}}) = 0$ and $\sigma^2(\tilde{\mathbf{v}}) = 0$. Replacing these values in the depressed quartic equation (30) causes the coefficients p and r to become constants, and coefficient q to vanish (also the orientation constraint vanishes). The depressed quartic thus transforms into a bi-quadratic:

$$\lambda^4 - \frac{32}{3}\lambda^2 + \frac{256}{9} = 0, \tag{A.1}$$

whose solutions are:

$$|\lambda| = \lambda_o. \tag{A.2}$$

We now turn to the reverse implication: $S_1 \Leftarrow A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = \lambda_o$. We substitute $|\lambda| = \lambda_o$ in equation (28), giving:

$$s \operatorname{sign}(A^*(\tilde{\mathbf{v}})) \operatorname{sign}(\lambda)\lambda_o A(\tilde{\mathbf{v}}) - \sigma^2(\tilde{\mathbf{v}}) = 0. \tag{A.3}$$

Since $A(\tilde{\mathbf{v}}) = 0$, then the only solution that satisfies equation (A.3) for any given value of $s \operatorname{sign}(A^*(\tilde{\mathbf{v}})) \operatorname{sign}(\lambda)$ is with $\sigma^2(\tilde{\mathbf{v}}) = 0$, which implies that the input triangle is collapsed into a single point, hence to S_1. □

Proof of Lemma 2. We start with the forward implication: $S_2 \Rightarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -\lambda_o$. In S_2, $\tilde{\mathbf{v}}$ represents an equilateral triangle and orientation inversion. This implies $A(\tilde{\mathbf{v}}) \neq 0$ and $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = -s$.

First, we show that λ is scale invariant; the invariance to rotation and translation is trivial. In Case I (when $|\lambda| \neq \lambda_o$), λ has a varying value, depending on the inputs. Specifically, it is resolved from the depressed quartic equation (30). Inspecting this quartic, we trivially see that its coefficients are scale, rotation and translation invariant. This proves that lambda, which is a root of this polynomial, is also scale, translation and rotation invariant. In Case II, (when $|\lambda| = \lambda_o$), λ has a fixed absolute value. In setting S_1, we have shown in Section 2.6 that $\operatorname{sign}(\lambda) = -s$, which is thus scale invariant. In settings S_2 and S_3, we have shown that $\operatorname{sign}(\lambda) = s \operatorname{sign}(A^*(\tilde{\mathbf{v}}))$ and since $\operatorname{sign}(A^*(\tilde{\mathbf{v}}))$ is not affected by positive scaling, rotation or translation, then S_2 and S_3 are also scale invariant.

Since λ is rotation, translation and scale invariant, we can safely perform a similarity transformation to $\tilde{\mathbf{v}}$ to simplify the problem. We bring one of its vertices to the origin and another one to the x-axis, scaled to normalise their distance, giving $\tilde{\mathbf{v}}' = [0, 0, 1, 0, 1/2, \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\sqrt{3}/2]^\top$ where $\operatorname{sign}(A^*(\tilde{\mathbf{v}}))$ determines the orientation of the triangle. We obtain $A^*(\tilde{\mathbf{v}}') = \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\frac{\sqrt{3}}{4}$ and $\sigma^2(\tilde{\mathbf{v}}') = 1$, thus the coefficients of the depressed quartic equation (30) become:

$$p = -\frac{32A_o - 4\sqrt{3}}{3A_o} \tag{A.4}$$

$$q = \frac{32}{3A_o} \tag{A.5}$$

$$r = \frac{256A_o + 64\sqrt{3}}{9A_o}. \tag{A.6}$$

Substituting these coefficients in Cardano's formula we obtain:

$$Q_1 = -\left(\frac{32A_o + 2\sqrt{3}}{9A_0}\right)^2 \tag{A.7}$$

$$Q_2 = \left(\frac{32A_o + 2\sqrt{3}}{9A_0}\right)^3, \tag{A.8}$$

making $\sqrt{Q_1^3 + Q_2^2} = 0$ and the real root α_o of Cardano's resolvent cubic to become:

$$\alpha_o = 2\left(\frac{32A_o + 2\sqrt{3}}{9A_o}\right) + \frac{32A_o - 4\sqrt{3}}{9A_o} = \frac{32}{3}. \tag{A.9}$$

We finally use α_o to extract the roots of the depressed quartic:

$$\lambda = \frac{s_1\sqrt{2}\lambda_o + s_2\sqrt{-\lambda_o\left(\frac{s_1+1}{A_o}\right)}}{\sqrt{2}}, \tag{A.10}$$

where $s_1, s_2 \in \{-1, 1\}$. We thus have the following roots:

$$\lambda \in \left\{-\lambda_o, -\lambda_o, \lambda_o - i\sqrt{\frac{\lambda_o}{A_o}}, \lambda_o + i\sqrt{\frac{\lambda_o}{A_o}}\right\}. \tag{A.11}$$

Considering only the real roots we have $\lambda = -\lambda_o$.

We now turn to the reverse implication: $S_2 \Leftarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -\lambda_o$. We substitute $\lambda = -\lambda_o$ in equation (28), giving:

$$-s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\lambda_o A(\tilde{\mathbf{v}}) - \sigma^2(\tilde{\mathbf{v}}) = 0. \tag{A.12}$$

Since $A(\tilde{\mathbf{v}}) \neq 0$ and $\sigma^2(\tilde{\mathbf{v}}) > 0$ then equation (A.12) can only be solved when $\mathrm{sign}(A^*(\tilde{\mathbf{v}})) = -s$. We perform the same similarity transformation used at the beginning of the proof to the unknown input triangle $\tilde{\mathbf{v}}$ giving $\tilde{\mathbf{v}}' = [0, 0, 1, 0, \tilde{x}'_c, \tilde{y}'_c]^\top$ where one of the vertices remains unknown. We then have $A^*(\tilde{\mathbf{v}}') = \frac{\tilde{y}'_c}{2}$ or $\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))A(\tilde{\mathbf{v}}') = \frac{\tilde{y}'_c}{2}$ and $\sigma^2(\tilde{\mathbf{v}}') = \frac{2}{3}(\tilde{x}'^2_c + \tilde{y}'^2_c - \tilde{x}'_c + 1)$ which we substitute in equation (A.12) and obtain:

$$\tilde{x}'^2_c + \tilde{y}'^2_c - \tilde{x}'_c + \sqrt{3}s\tilde{y}'_c + 1 = 0, \tag{A.13}$$

which we rewrite as:

$$\left(\tilde{x}'_c - \frac{1}{2}\right)^2 + \left(\tilde{y}'_c + \frac{\sqrt{3}}{2}s\right)^2 = 0. \tag{A.14}$$

This is the equation of a single point, making $\tilde{\mathbf{v}}'$ an equilateral triangle $\tilde{\mathbf{v}}' = [0, 0, 1, 0, 1/2, -\sqrt{3}s/2]^\top$, where s determines the orientation of the triangle. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\tilde{\mathbf{v}}$ is also an equilateral triangle when $\text{sign}(A^*(\tilde{\mathbf{v}})) = -s$, which corresponds to S_2. □

Proof of Lemma 3. In S_3, $\tilde{\mathbf{v}}$ represents an equilateral triangle with $A(\tilde{\mathbf{v}})/4 \geq A_o$ and no orientation inversion. This implies $A^*(\tilde{\mathbf{v}}) \neq 0$ and $\text{sign}(A^*(\tilde{\mathbf{v}})) = s$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 2, thus the coefficients of the depressed quartic equation (30) become:

$$p = -\frac{32A_o + 4\sqrt{3}}{3A_o} \tag{A.15}$$

$$q = \frac{32}{3A_o} \tag{A.16}$$

$$r = \frac{256A_o - 64\sqrt{3}}{9A_o}. \tag{A.17}$$

Substituting these coefficients in Cardano's formula we obtain:

$$Q_1 = -\left(\frac{32A_o - 4}{9A_0}\right)^2 \tag{A.18}$$

$$Q_2 = -\left(\frac{32A_o - 4}{9A_0}\right)^3, \tag{A.19}$$

making $\sqrt{Q_1^3 + Q_2^2} = 0$. After some factoring we obtain the real root α_o of Cardano's resolvent cubic as:

$$\alpha_o = -2\sqrt[3]{\left(\frac{32A_o - 8sA^*(\tilde{\mathbf{v}}')}{3A_o}\right)^3} + \frac{32A_o + 8sA^*(\tilde{\mathbf{v}}')}{3A_o}. \tag{A.20}$$

Since $\text{sign}(A^*(\tilde{\mathbf{v}})) = s$ and $A^*(\tilde{\mathbf{v}}') = \text{sign}(A^*(\tilde{\mathbf{v}}))A(\tilde{\mathbf{v}}')$, then $\alpha_o \in \mathbb{R}$ only when $A(\tilde{\mathbf{v}}')/8 \geq A_o$. Under this condition, we obtain $\alpha_o = 32$. Substituting in equation (37) we obtain:

$$\lambda = \frac{s_1\sqrt{2}\lambda_o + s_2\sqrt{\lambda_o\left(\frac{s_1 - 1}{A_o}\right)}}{\sqrt{2}} \tag{A.21}$$

where $s_1, s_2 \in \{-1, 1\}$. We thus have the following roots:

$$\lambda \in \left\{ -\lambda_o - \sqrt{\frac{\lambda_o}{A_o}}, -\lambda_o + \sqrt{\frac{\lambda_o}{A_o}}, \lambda_o, \lambda_o \right\}. \tag{A.22}$$

Thus, we have two roots where $|\lambda| \neq \lambda_o$ (which correspond to solutions for Case I) and there exists at least one solution $\lambda = \lambda_o$ for S_3 (which correspond to the solution of Case II). □

Proof of Lemma 4. We substitute $\lambda = \lambda_o$ in equation (28), giving:

$$s \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\lambda_o A(\tilde{\mathbf{v}}) - \sigma^2(\tilde{\mathbf{v}}) = 0. \tag{A.23}$$

Since $A(\tilde{\mathbf{v}}) \neq 0$ then equation (A.23) can only be solved when $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 2, substitute $\operatorname{sign}(A^*(\tilde{\mathbf{v}}'))A(\tilde{\mathbf{v}}')$ and $\sigma^2(\tilde{\mathbf{v}}')$ in equation (A.23) and obtain:

$$\tilde{x}_c'^2 + \tilde{y}_c'^2 - \tilde{x}_c' + \sqrt{3}s\tilde{y}_c' + 1 = 0, \tag{A.24}$$

which we rewrite as:

$$\left(\tilde{x}_c' - \frac{1}{2}\right)^2 + \left(\tilde{y}_c' + \frac{\sqrt{3}}{2}s\right)^2 = 0. \tag{A.25}$$

This is the equation of a single point making $\tilde{\mathbf{v}}'$ an equilateral triangle $\tilde{\mathbf{v}}' = \left[0, 0, 1, 0, 1/2, -\sqrt{3}s/2\right]^\top$ where s determines the orientation of the triangle. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\tilde{\mathbf{v}}$ is also an equilateral triangle when $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$ for any value $A(\tilde{\mathbf{v}})$ (including $A(\tilde{\mathbf{v}}')/4 \geq A_o$) which corresponds to S_3. □

Proof of Lemma 5. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 2 and obtain $A^*(\tilde{\mathbf{v}}') = \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\frac{\sqrt{3}}{4} = \frac{\operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o}$. We relax the condition of S_3 where $\frac{A(\tilde{\mathbf{v}}')}{4} \geq A_o$ by reformulating $A_o = \frac{1}{z\lambda_o}$ where z is a scaling factor $z > 0 \in \mathbb{R}$. We substitute this in equation (A.22) and extract the roots :

$$\lambda = \left[-\lambda_o - \lambda_o\sqrt{z}, -\lambda_o + \lambda_o\sqrt{z}, \lambda_o, \lambda_o\right]^\top. \tag{A.26}$$

We start with the solution given by Case I where $|\lambda| \neq \lambda_o$. We compact both values as $\lambda = -\lambda_o + s_3\lambda_o\sqrt{z}$ where $s_3 \in \{-1, 1\}$. We substitute this in equation (22) and obtain:

$$\mathbf{v}' = \begin{bmatrix} \frac{\sqrt{z}-s_3}{2\sqrt{z}} & \frac{\sqrt{3}(\sqrt{z}-s_3)}{6\sqrt{z}} & \frac{\sqrt{z}+s_3}{2\sqrt{z}} & \frac{\sqrt{3}(\sqrt{z}-s_3)}{6\sqrt{z}} & \frac{1}{2} & \frac{\sqrt{3}(\sqrt{z}+2s_3)}{6\sqrt{z}} \end{bmatrix}^\top. \tag{A.27}$$

We calculate the cost of the solution of Case I by substituting this and $\tilde{\mathbf{v}}'$ in equation (3) and obtain:

$$\mathscr{C}_1(\mathbf{v}') = \frac{(\sqrt{z} - s_3)^2}{z} \tag{A.28}$$

Now we turn to the solution given by Case II where $\lambda = \lambda_o$. We calculate the basis vector $\mathbf{v}'_c$, the particular solution $\mathbf{v}'_p$ and substitute them in equation (49) and obtain:

$$\mathbf{v}' = \begin{bmatrix} \frac{1}{4} - \frac{\sqrt{3}\sqrt{z-4}}{12\sqrt{z}} \\ \frac{\sqrt{3}}{12} - \frac{\sqrt{z-4}}{4\sqrt{z}} \\ \frac{3}{4} - \frac{\sqrt{3}\sqrt{z-4}}{12\sqrt{z}} \\ \frac{\sqrt{3}}{12} + \frac{\sqrt{z-4}}{4\sqrt{z}} \\ \frac{1}{2} + \frac{\sqrt{3}\sqrt{z-4}}{6\sqrt{z}} \\ \frac{\sqrt{3}}{3} \end{bmatrix}. \tag{A.29}$$

We calculate the cost of the solution of Case II by substituting this and $\tilde{\mathbf{v}}'$ in equation (3) and obtain:

$$\mathscr{C}_2(\mathbf{v}') = \frac{1}{2} - \frac{1}{z} \tag{A.30}$$

We compare the cost of both solutions $\mathscr{C}_1(\mathbf{v}') \geq \mathscr{C}_2(\mathbf{v}')$ and obtain:

$$\frac{(\sqrt{z} - s_3)^2}{z} \geq \frac{1}{2} - \frac{1}{z} \tag{A.31}$$

After some minor manipulations we obtain:

$$\frac{z - 4s_3\sqrt{z} + 6}{2z} \geq 0 \tag{A.32}$$

We substitute $\sqrt{z} = a$ where $a > 0 \in \mathbb{R}$ and obtain the quadratic expression:

$$\frac{a^2 - 4s_3 a + 6}{2a^2} \geq 0 \tag{A.33}$$

which represents an upward opening parabola which is always positive for any value of a and thus z. This implies that $\mathscr{C}_1(\mathbf{v}') \geq \mathscr{C}_2(\mathbf{v}')$ for any value z including $z > 4$. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\mathscr{C}_1(\mathbf{v}) \geq \mathscr{C}_2(\mathbf{v})$. This means that in S_3 the solution provided by case II has the lowest cost, thus is the optimal solution. □

Proof of Lemma 6. We start with the forward implication: $\mathbf{v}_p = 0 \Rightarrow \mathbf{v}'$ is an equilateral triangle. From equation (43), we have that $\mathbf{v}_p = 0$ when $\tilde{\mathbf{v}}' = 0$ (trivial solution) or when $\tilde{\mathbf{v}}' = 0$ is in the kernel of $X^\dagger$. From the properties of the pseudo-inverse we have that $\ker(X^\dagger) = \ker(X^\top)$. Since X is symmetric then $\ker(X^\dagger) = \ker(X) = \mathbf{v}_h$, as given in equation (42). Recall from Section 2.6 that the system $X\mathbf{v}' = \tilde{\mathbf{v}}'$ is solvable if and only if $\tilde{\mathbf{v}}'$ represents an equilateral triangle of orientation $\operatorname{sign}(A^*(\tilde{\mathbf{v}}')) = s \operatorname{sign}(\lambda)$ or colocated vertices. Still from Section 2.6, we have that $\mathbf{v}_h$ is an equilateral triangle of orientation $\operatorname{sign}(A^*(\mathbf{v}_h)) = -s \operatorname{sign}(\lambda)$, which contradicts the previous statement. This leaves us with $\mathbf{v}_p = 0$ only when $\tilde{\mathbf{v}}' = 0$, which represents a set of colocated points centred in the origin. We then calculate $\mathbf{v}'$ from equation (45) and obtain:

$$\mathbf{v}' = \begin{bmatrix} -\frac{\beta_1}{2} + \frac{2\beta_2 \operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o} \\ -\frac{2\beta_1 \operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o} - \frac{\beta_2}{2} \\ -\frac{\beta_1}{2} - \frac{2\beta_2 \operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o} \\ \frac{2\beta_1 \operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o} - \frac{\beta_2}{2} \\ \beta_1 \\ \beta_2 \end{bmatrix}. \tag{A.34}$$

We can then easily show that the inter-vertex distances are equal, and thus that $\mathbf{v}'$ is an equilateral triangle:

$$\begin{aligned} D_{ab} =& \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2} = \sqrt{3\beta_1^2 + 3\beta_2^2} \\ D_{bc} =& \sqrt{(x'_b - x'_c)^2 + (y'_b - y'_c)^2} = \sqrt{3\beta_1^2 + 3\beta_2^2} \\ D_{ac} =& \sqrt{(x'_a - x'_c)^2 + (y'_a - y'_c)^2} = \sqrt{3\beta_1^2 + 3\beta_2^2}. \end{aligned}$$

We now turn to the reverse implication: $\mathbf{v}_p = 0 \not\Leftarrow \mathbf{v}'$ is an equilateral triangle, for which we simply provide a counterexample to the positive implication. We use the equilateral triangle $\tilde{\mathbf{v}}' = [0, 0, 1, 0, 1/2, \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\sqrt{3}/2]^\top$ where $A(\tilde{\mathbf{v}}')/4 = A_o$ and $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$. We calculate its particular solution with equation (43) and obtain:

$$\mathbf{v}_p = \begin{bmatrix} \frac{1}{4} \\ \frac{\operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} \\ \frac{3}{4} \\ \frac{\operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} \\ \frac{1}{2} \\ \frac{4 \operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} \end{bmatrix}, \tag{A.35}$$

which implies that $\mathbf{v}_p \neq 0$. We substitute $A(\tilde{\mathbf{v}}')/4 = A_o$ and $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$ in the constraint equation (45) and obtain $\beta_1^2 = -\beta_2^2$. This implies that the constraint is fulfilled when $\beta_1 = \beta_2 = 0$ thus $\mathbf{v}' = \mathbf{v}_p$. We easily show that the inter-vertex distances are equal:

$$\begin{aligned} D_{ab} =& \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2} = \frac{1}{2} \\ D_{bc} =& \sqrt{(x'_b - x'_c)^2 + (y'_b - y'_c)^2} = \frac{1}{2} \\ D_{ac} =& \sqrt{(x'_a - x'_c)^2 + (y'_a - y'_c)^2} = \frac{1}{2}, \end{aligned}$$

thus $\mathbf{v}$ is an equilateral triangle despite $\mathbf{v}_p \neq 0$. □

Proof of Lemma 7. Our proof shows that the cost is invariant to the value chosen for angle θ. We start by translating the coordinate system to bring the input's centroid to the origin as $\tilde{\mathbf{v}}' = \tilde{\mathbf{v}} - \bar{\mathbf{v}}$, which also translates the unknown vertices to $\mathbf{v}' = \mathbf{v} - \bar{\mathbf{v}}$. Thus, the cost becomes the translated least-squares displacement cost:

$$\mathscr{C}'(\mathbf{v}') = \|\mathbf{v}' - \tilde{\mathbf{v}}'\|^2. \tag{A.36}$$

We then continue with the case where $\tilde{\mathbf{v}}$ is a single point. This implies $A(\tilde{\mathbf{v}}) = 0$ and $k = -1$. The particular solution $\mathbf{v}'_p$ can be verified to vanish, from equation (43). We calculate the basis vector $\mathbf{v}'_c$ from equation (48), leading, from equation (49), to:

$$\mathbf{v}' = \begin{bmatrix} \frac{2\phi \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\sin(\theta)}{\lambda_o} - \frac{\phi\cos(\theta)}{2} \\ -\frac{\phi\sin(\theta)}{2} - \frac{2\phi \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\cos(\theta)}{\lambda_o} \\ -\frac{2\phi \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\sin(\theta)}{\lambda_o} - \frac{\phi\cos(\theta)}{2} \\ -\frac{\phi\sin(\theta)}{2} + \frac{2\phi \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\cos(\theta)}{\lambda_o} \\ \phi\cos(\theta) \\ \phi\sin(\theta) \end{bmatrix}. \tag{A.37}$$

We calculate the cost by substituting $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{v}}'$ in equation (A.36):

$$\mathscr{C}'(\mathbf{v}') = A_o\lambda_o(\sin^2(\theta) + \cos^2(\theta)), \tag{A.38}$$

which simplifies to $\mathscr{C}'(\mathbf{v}') = A_o\lambda_o$ and is thus independent of θ.

We continue with the case where $\tilde{\mathbf{v}}$ is an equilateral triangle. This implies that $A(\tilde{\mathbf{v}}) \neq 0$ and $k = s \operatorname{sign}(A^*(\tilde{\mathbf{v}}))$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 2 and obtain $A^*(\tilde{\mathbf{v}}') = \operatorname{sign}(A^*(\tilde{\mathbf{v}}))\frac{\sqrt{3}}{4} = \frac{\operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{\lambda_o}$. We calculate the basis vector $\mathbf{v}'_c$ from equation (48), the particular solution $\mathbf{v}'_p$ from equation (43) and substitute them in equation (49), leading to:

$$\mathbf{v}' = \begin{bmatrix} \frac{1}{4} + \frac{2\phi\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\sin(\theta)}{\lambda_o} - \frac{\phi\cos(\theta)}{2} \\ \frac{\mathrm{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} - \frac{\phi\sin(\theta)}{2} - \frac{2\phi\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\cos(\theta)}{\lambda_o} \\ \frac{3}{4} - \frac{2\phi\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\sin(\theta)}{\lambda_o} - \frac{\phi\cos(\theta)}{2} \\ \frac{\mathrm{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} - \frac{\phi\sin(\theta)}{2} + \frac{2\phi\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\cos(\theta)}{\lambda_o} \\ \frac{1}{2} + \phi\cos(\theta) \\ \frac{4\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))}{3\lambda_o} + \phi\sin(\theta) \end{bmatrix}. \tag{A.39}$$

We calculate the cost by substituting $\mathbf{v}'$ and $\tilde{\mathbf{v}}'$ in equation (A.36):

$$\mathscr{C}'(\mathbf{v}') = \frac{1}{4} + \frac{\sin^2(\theta) + \cos^2(\theta)}{4} - s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{A_o(\sin^2(\theta) + \cos^2(\theta))}{\lambda_o}, \tag{A.40}$$

which simplifies to $\mathscr{C}'(\mathbf{v}') = \frac{1}{2} - s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))\frac{A_o}{\lambda_o}$ and is thus independent of θ. □

Appendix B. Optimal Triangle Projection with Prescribed Area

In OTPPA, only the prescribed area needs to be preserved. This means that a solution may freely choose the orientation which minimises the cost, as long as the area constraint is satisfied. Consequently, we expect that OTPPA has a larger set of local extrema than OTPPAO and hence more candidate algebraic solutions. Specifically, OTPPA is stated as:

$$\min_{\mathbf{v}\in\mathbb{R}^6} \mathscr{C}(\mathbf{v}) \quad \text{s.t.} \quad f(\mathbf{v}) = 0. \tag{B.1}$$

The area constraint in OTPPA is technically more complex to handle than the signed area constraint in OTPPAO, because it involves an absolute value. Fortunately, a solution may be obtained by exploiting a reformulation in terms of two rounds of OTPPAO. Similarly to OTPPAO, we start by replacing $A(\mathbf{v})$ by $A^*(\mathbf{v})$ in the area constraint $f(\mathbf{v})$, expressing f as the disjunction of two cases:

$$f(\mathbf{v}) = 0 \quad \Leftrightarrow \quad \big(f^+(\mathbf{v}) = 0\big) \vee \big(f^-(\mathbf{v}) = 0\big) \quad \text{with} \tag{B.2}$$

$$f^+(\mathbf{v}) = A^*(\mathbf{v}) - A_o \tag{B.3}$$

$$f^-(\mathbf{v}) = -A^*(\mathbf{v}) - A_o. \tag{B.4}$$

We seek a solution which satisfies either f^+ or f^-. This can be achieved by solving OTPPAO for $s = 1$ and $s = -1$, and simply selecting the minimal cost solution a posteriori. We present the numerically robust procedure in Algorithm 7.

Algorithm 7 Optimal Triangle Projection with Prescribed Area

Input: $\tilde{\mathbf{v}}$ - input vertices, A_o - prescribed area, s - prescribed orientation, E - area error tolerance

Output: $\mathbf{v}_o$ - optimal triangle, $\mathcal{v}_1$ - Case I triangle set, $\mathcal{v}_2$ - Case II triangle set, $\mathcal{v}_c, \mathcal{v}_t$ - Case II basis and translations sets

```
function OTPPA($\tilde{\mathbf{v}}, A_o, s, E = 10^{-3}$)
    $\mathcal{v}_1^- \leftarrow$ SolveCase1($\tilde{\mathbf{v}}, A_o, -1, E$)    ▷ Compute Case I solutions
    $\mathcal{v}_1^+ \leftarrow$ SolveCase1($\tilde{\mathbf{v}}, A_o, 1, E$)
    $(\mathbf{v}_2^-, \mathbf{v}_c^-, \mathbf{v}_t^-) \leftarrow$ SolveCase2($\tilde{\mathbf{v}}, A_o, -1, E$)    ▷ Compute Case II solutions
    $(\mathbf{v}_2^+, \mathbf{v}_c^+, \mathbf{v}_t^+) \leftarrow$ SolveCase2($\tilde{\mathbf{v}}, A_o, 1, E$)
    $\mathcal{v}_1 \leftarrow \mathcal{v}_1^- \cup \mathcal{v}_1^+$    ▷ Add the vertices to the solution set
    $\mathcal{v}_2 \leftarrow \{\mathbf{v}_2^-\} \cup \{\mathbf{v}_2^+\}$
    $\mathcal{v}_c \leftarrow \{\mathbf{v}_c^-\} \cup \{\mathbf{v}_c^+\}$
    $\mathcal{v}_t \leftarrow \{\mathbf{v}_t^-\} \cup \{\mathbf{v}_t^+\}$
    $\mathbf{v}_o \leftarrow$ FindTriangleOfMinimalCost($\mathcal{v}_1 \cup \mathcal{v}_2$)    ▷ Select the optimal solution
    return $\mathbf{v}_o, \mathcal{v}_1, \mathcal{v}_2, \mathcal{v}_c, \mathcal{v}_t$
end function
```

Appendix C. Ferrari's Method for the Depressed Quartic Roots

We give Ferrari's method for solving the depressed quartic equation $\lambda^4 - p\lambda^2 + q\lambda + r = 0$ ([13]).

Lemma 8. *If $\lambda^4 + p\lambda^2 + q\lambda + r = 0$ and $q \neq 0$ then there exists an $\alpha_o \neq 0$ such that $\lambda = \frac{s_1\sqrt{2\alpha_o}+s_2\sqrt{-\left(2p+2\alpha_o+s_1\frac{2q}{\sqrt{2\alpha_o}}\right)}}{2}$, where $s_1, s_2 \in \{-1, 1\}$.*

Proof of Lemma 8. First, one adds and subtracts $\frac{p^2}{4}$ to the depressed quartic equation and rewrites it as:

$$\left(\lambda^2 + \frac{p}{2}\right)^2 = -q\lambda - r + \frac{p^2}{4}. \tag{C.1}$$

The equality to the original quartic can be verified by simple expansion. One then introduces a variable factor α into the left-hand side by adding $2\lambda^2\alpha + p\alpha + \alpha^2$ to both sides. Grouping the coefficients by powers of λ in the right-hand side gives:

$$\left(\lambda^2 + \frac{p}{2} + \alpha\right)^2 = 2\alpha\lambda^2 - q\lambda + \left(\alpha^2 + \alpha p + \frac{p^2}{4} - r\right). \tag{C.2}$$

A quadratic expression $ax^2 + bx + c$ is considered a perfect square when its discriminant $b^2 - 4ac = 0$ vanishes, allowing one to rewrite it as $(\sqrt{a}x + \sqrt{c})^2$. We use this idea to choose a value for α such that the bracketed expression in the right-hand side of equation (C.2), which is a quadratic in λ, becomes a perfect square. Specifically, vanishing the discriminant gives:

$$q^2 - 8\alpha\left(\alpha^2 + \alpha p + \frac{p^2}{4} - r\right) = 0. \tag{C.3}$$

Upon expanding, it forms a cubic equation in α, called the resolvent cubic of the quartic equation:

$$8\alpha^3 + 8p\alpha^2 + (2p^2 - 8r)\alpha - q^2 = 0. \tag{C.4}$$

This equation implies $\alpha \neq 0$. Indeed, $\alpha = 0$ would imply $q = 0$, contradicting our hypothesis $q \neq 0$. A real root $\alpha_o \neq 0$ is obtained from Cardano's formula, given in section Appendix D. Substituting in equation (C.2), we obtain:

$$\left(\lambda^2 + \frac{p}{2} + \alpha_o\right)^2 = \left(\lambda\sqrt{2\alpha_o} - \frac{q}{2\sqrt{2\alpha_o}}\right)^2. \tag{C.5}$$

This equation is of the form $M^2 = N^2$, which can be rearranged as $M^2 - N^2 = 0$ or $(M + N)(M - N) = 0$:

$$\left(\lambda^2 + \frac{p}{2} + \alpha_o + \lambda\sqrt{2\alpha_o} - \frac{q}{2\sqrt{2\alpha_o}}\right)\left(\lambda^2 + \frac{p}{2} + \alpha_o - \lambda\sqrt{2\alpha_o} + \frac{q}{2\sqrt{2\alpha_o}}\right) = 0. \tag{C.6}$$

This is easily solved by applying the quadratic formula to each factor, leading to:

$$\lambda = \frac{s_1\sqrt{\alpha_o} + s_2\sqrt{-\left(p + \alpha_o + s_1\frac{q}{\sqrt{2\alpha_o}}\right)}}{\sqrt{2}}, \tag{C.7}$$

where $s_1, s_2 \in \{-1, 1\}$. □

Appendix D. Cardano's Method for the Cubic Roots

We consider the cubic equation:

$$ax^3 + bx^2 + cx + d = 0 \quad \text{with} \quad a \neq 0.$$

Its solutions are:

$$\begin{aligned}
x_1 &= S_1 + S_2 - \frac{b}{3a} \\
x_2 &= -\frac{S_1 + S_2}{2} - \frac{b}{3a} + \frac{i\sqrt{3}}{2}(S_1 - S_2) \\
x_3 &= -\frac{S_1 + S_2}{2} - \frac{b}{3a} - \frac{i\sqrt{3}}{2}(S_1 - S_2),
\end{aligned}$$

where:

$$\begin{aligned}
S_1 &= \sqrt[3]{Q_2 + \sqrt{Q_1^3 + Q_2^2}} \\
S_2 &= \sqrt[3]{Q_2 - \sqrt{Q_1^3 + Q_2^2}} \\
Q_1 &= \frac{3ac - b^2}{9a^2} \\
Q_2 &= \frac{9abc - 27a^2d - 2b^3}{54a^3},
\end{aligned}$$

and $D = Q_1^3 + Q_2^2$ is the discriminant of the equation. For $a, b, c, d \in \mathbb{R}$, three cases can occur:

$$\begin{aligned}
(1):&\quad \text{if}\, D > 0,\ \text{one root is real and two are complex conjugates} \\
(2):&\quad \text{if}\, D = 0,\ \text{all roots are real, and at least two are equal} \\
(3):&\quad \text{if}\, D < 0,\ \text{all roots are real and unequal.}
\end{aligned}$$

Appendix E. Formulation for Restricted Cases

Our original formulation assumes that the three triangle vertices are free to move. However, there exist cases when one or two of the vertices are fixed. Typically, this occurs for triangles lying on the domain boundary in mesh editing. We here adapt the proposed optimal projection formulation to these cases.

Appendix E.1. One Fixed Vertex

Appendix E.1.1. A Two Case Formulation

We assume that v_c is fixed. Thus, we have $\mathbf{v} = [\mathbf{u}, \tilde{x}_c, \tilde{y}_c]$ and $\tilde{\mathbf{v}} = [\tilde{\mathbf{u}}, \tilde{x}_c, \tilde{y}_c]$, where the moving vertices are represented by $\mathbf{u} = [x_a, y_a, x_b, y_b] \in \mathbb{R}^4$ and the corresponding input vertices by $\tilde{\mathbf{u}} = [\tilde{x}_a, \tilde{y}_a, \tilde{x}_b, \tilde{y}_b] \in \mathbb{R}^4$. We take $\frac{\partial \mathscr{L}}{\partial \mathbf{u}} = 0$ which is formed by the first four equalities of equation (12), which we rewrite in matrix form $X\mathbf{u} = \mathbf{b}$ as:

$$\begin{bmatrix} 4 & 0 & 0 & s\lambda \\ 0 & 4 & -s\lambda & 0 \\ 0 & -s\lambda & 4 & 0 \\ s\lambda & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} x_a \\ y_a \\ x_b \\ y_b \end{bmatrix} = 4 \begin{bmatrix} \tilde{x}_a + s\frac{\lambda}{4}(\tilde{y}_c) \\ \tilde{y}_a - s\frac{\lambda}{4}(\tilde{x}_c) \\ \tilde{x}_b - s\frac{\lambda}{4}(\tilde{y}_c) \\ \tilde{y}_b + s\frac{\lambda}{4}(\tilde{x}_c) \end{bmatrix}. \tag{E.1}$$

We check the invertibility of X from its determinant:

$$\det(X) = (\lambda^2 - 16)^2. \tag{E.2}$$

We thus have:

$$\det(X) = 0 \quad \Leftrightarrow \quad |\lambda| = 4. \tag{E.3}$$

We will see that the particular case of $\det(X) = 0$ may occur in practice. We thus solve system (E.1) with two cases. In Case I, which is the most general, we have $|\lambda| \neq 4$. In Case II, we have $|\lambda| = 4$. Similarly to the general case in Section 2.3, the following Lemmas establish the relationship between the Lagrange multiplier and the linear deficiency of the input vertices.

Proposition 2. *Most settings (input vertices $\tilde{\mathbf{v}}$, prescribed area A_o and orientation s) correspond to F_o and fall in Case I. Exceptions handled with Case II are:*

- *F_1: $\tilde{\mathbf{v}}$ is a single point*
- *F_2: $\tilde{\mathbf{v}}$ is a right isosceles triangle and* $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = -s$
- *F_3: $\tilde{\mathbf{v}}$ is a right isosceles triangle, $A(\tilde{\mathbf{v}})/4 \geq A_o$ and* $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$

The proof of Proposition 2 is based on the following five Lemmas.

Lemma 9. *$F_1 \iff A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = 4$.*

Lemma 10. *$F_2 \iff A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -4$.*

Lemma 11. *$F_3 \Rightarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda \in \left\{4, -4 + 2\sqrt{\frac{2}{A_o}}, -4 - 2\sqrt{\frac{2}{A_o}}\right\}$.*

Lemma 12. *$F_3 \Leftarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = 4$.*

Lemma 13. *$\lambda = 4$ leads to the optimal solution for F_3.*

The proofs of these Lemmas are given in Appendix E.1.4. It is important to clarify for Proposition 2, that in the right isosceles triangle, the fixed vertex corresponds to the one opposite to the triangle's hypotenuse.

Proof of Proposition 2. We recall that Case I occurs for $|\lambda| \neq 4$ and Case II for $|\lambda| = 4$. Lemmas 9, 10 and 12 show that F_1, F_2 and F_3 are the only possible settings corresponding to $|\lambda| = 4$, hence possibly to Case II. This proves that Case I is the general case. Lemmas 9 and 10 then trivially prove that F_1 and F_2 are handled by Case II. Finally, Lemmas 11 and 13 prove that F_3 is also handled by Case II. □

Appendix E.1.2. Case I

This case occurs for $|\lambda| \neq 4$. In other words, this is the case where at least one of the initial vertices in $\tilde{\mathbf{u}}$ is different from the other two and where the rank of the input matrix M in equation (18) is $\text{rank}(M) > 1$. Similarly to Case I for three moving vertices, the problem is reformulated as a depressed quartic polynomial and the roots of this polynomial are found using Ferrari's method. We start by multiplying equation (E.1) by the adjugate X^* of X and obtain:

$$\det(X)\mathbf{u} = X^*\mathbf{b}, \tag{E.4}$$

where the adjugate is:

$$X^* = \delta Y = \lambda^2 - 16 \begin{bmatrix} -4 & 0 & 0 & s\lambda \\ 0 & -4 & -s\lambda & 0 \\ 0 & -s\lambda & -4 & 0 \\ s\lambda & 0 & 0 & -4 \end{bmatrix}, \tag{E.5}$$

with $\delta = \lambda^2 - 16$ and $Y \in \mathbb{R}^{4\times4}$. We note that $\det(X) = \delta^2$. We substitute this and equation (E.5) in equation (E.4) and obtain:

$$\delta\mathbf{u} = Y\mathbf{b}. \tag{E.6}$$

We observe that the signed area $A^*(\delta\mathbf{v}) = \delta^2 A^*(\mathbf{v})$. Also that $\delta\mathbf{v} = [\mathbf{u}, \delta\tilde{x}_c, \delta\tilde{y}_c]$. Thus, we calculate the signed area of $\delta\mathbf{v}$ and obtain after some minor expanding:

$$\delta^2 A^*(\mathbf{v}) = a_2\lambda^2 + a_1\lambda + a_o, \tag{E.7}$$

where:

$$a_0 = 128((\tilde{x}_a - \tilde{x}_c)(\tilde{y}_b - \tilde{y}_a) - (\tilde{x}_a - \tilde{x}_b)(\tilde{y}_c - \tilde{y}_a)) \tag{E.8}$$

$$a_1 = -32\left(\tilde{x}_a^2 + \tilde{x}_b^2 + 2\tilde{x}_c^2 + \tilde{y}_a^2 + \tilde{y}_b^2 + 2\tilde{y}_c^2\right) \tag{E.9}$$

$$+ 64(\tilde{x}_a\tilde{x}_b + \tilde{x}_a\tilde{x}_c + \tilde{x}_b\tilde{x}_c + \tilde{y}_a\tilde{y}_b + \tilde{y}_a\tilde{y}_c + \tilde{y}_b\tilde{y}_c) \tag{E.10}$$

$$a_2 = 8((\tilde{x}_a - \tilde{x}_c)(\tilde{y}_b - \tilde{y}_a) - (\tilde{x}_a - \tilde{x}_b)(\tilde{y}_c - \tilde{y}_a)). \tag{E.11}$$

We rewrite these coefficients more compactly. Concretely, a_0 and a_2 contain the signed area $A^*(\tilde{\mathbf{v}})$ of the input vertices and a_1 contains the sum of the squared distances D_o of the two moving vertices to the fixed vertex:

$$D_o(\tilde{\mathbf{v}}) = (\tilde{x}_a - \tilde{x}_c)^2 + (\tilde{y}_a - \tilde{y}_c)^2 + (\tilde{x}_b - \tilde{x}_c)^2 + (\tilde{y}_b - \tilde{y}_c)^2. \tag{E.12}$$

We substitute equation (E.7) in the signed area constraint (8) multiplied by δ^2 and obtain:

$$sA^*(\mathbf{v}) - \delta^2 A_o = 0. \tag{E.13}$$

This way, the signed area only depends on the known initial vertices $\tilde{\mathbf{v}}$ and prescribed sign s. Because the signed area is quadratic in the vertices, and the vertices are quadratic rational in λ, the resulting equation is a quartic in λ:

$$A_o\lambda^4 - 16(2A_o + sA^*(\tilde{\mathbf{v}}))\lambda^2 + 32D_o(\tilde{\mathbf{v}})\lambda + 256(A_o - sA^*(\tilde{\mathbf{v}})) = 0. \tag{E.14}$$

This is a depressed quartic because it does not have a cubic term. We can thus rewrite it to the standard form by simply dividing by A_o, giving:

$$\lambda^4 + p\lambda^2 + q\lambda + r = 0, \tag{E.15}$$

with:

$$p = -\frac{16(2A_o + sA^*(\tilde{\mathbf{v}}))}{A_o} \tag{E.16}$$

$$q = \frac{32D_o(\tilde{\mathbf{v}})}{A_o} \tag{E.17}$$

$$r = \frac{256(A_o - sA^*(\tilde{\mathbf{v}}))}{A_o}. \tag{E.18}$$

This can be solved using Ferrari's method. We observe that when $\operatorname{rank}(M) = 2$, implying $A^*(\tilde{\mathbf{v}}) = 0$, coefficients p and r become constants. However, the equation remains a general depressed quartic which can be solved with our procedure.

Appendix E.1.3. Case II

Case II occurs for $|\lambda| = 4$. From Proposition 2, this means that the initial vertices $\tilde{\mathbf{v}}$ are colocated as $\tilde{v}_a = \tilde{v}_b = \tilde{v}_c$ or represent right isosceles triangles under conditions from Proposition 2. We show that the problem is represented by translated homogeneous and linearly dependent equations. We find their null space, particular solution and then a subset constrained by the prescribed area.

We first translate the coordinate system to bring the fixed vertex to the origin $\tilde{\mathbf{u}}' = \tilde{\mathbf{u}}' - v_c$ translating the unknown vertices to $\mathbf{u}' = \mathbf{u} - v_c$. Substituting $|\lambda| = 4$ in matrix X, we obtain:

$$4 \begin{bmatrix} 1 & 0 & 0 & s\,\mathrm{sign}(\lambda) \\ 0 & 1 & -s\,\mathrm{sign}(\lambda) & 0 \\ 0 & -s\,\mathrm{sign}(\lambda) & 1 & 0 \\ s\,\mathrm{sign}(\lambda) & 0 & 0 & 1 \end{bmatrix}. \tag{E.19}$$

This matrix has a rank of two and is thus non-invertible. Thus, $X\mathbf{u}' = \tilde{\mathbf{u}}'$ is solvable if and only if $\tilde{\mathbf{u}}'$ lies in the column space $C(X)$. The column space can be calculated by factoring X into its singular value decomposition (SVD) $X = W\Sigma W^\top$ and taking the first $\mathrm{rank}(X)$ columns of the unitary matrix W. For each value of $s\,\mathrm{sign}(\lambda)$, we have column spaces expressed as two-dimensional linear subspaces $\{\gamma_1\mathbf{w}_1^- + \gamma_2\mathbf{w}_2^-\}$ and $\{\gamma_1\mathbf{w}_1^+ + \gamma_2\mathbf{w}_2^+\}$ where $\gamma_1, \gamma_2 \in \mathbb{R}$ and with bases $\mathbf{w}_1^-, \mathbf{w}_2^- \in \mathbb{R}^4$ and $\mathbf{w}_1^+, \mathbf{w}_2^+ \in \mathbb{R}^4$ such that:

$$\begin{bmatrix} \mathbf{w}_1^- & \mathbf{w}_2^- \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}, \tag{E.20}$$

and:

$$\begin{bmatrix} \mathbf{w}_1^+ & \mathbf{w}_2^+ \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}, \tag{E.21}$$

We have that when we add a vertex located in the origin to $\mathbf{w}_1^-, \mathbf{w}_2^-, \mathbf{w}_1^+$ and $\mathbf{w}_2^+$, they represent right isosceles triangles of the same area of $\frac{1}{4}$ with orientation $s\,\mathrm{sign}(\lambda)$. Similarly, adding a vertex located in the origin to the linear combinations $\gamma_1\mathbf{w}_1^- + \gamma_2\mathbf{w}_2^-$ and $\gamma_1\mathbf{w}_1^+ + \gamma_2\mathbf{w}_2^+$ represent right isosceles triangles of any area and opposite orientations (or colocated points if $\gamma_1 = \gamma_2 = 0$). This shows that the system is solvable if and only if $\mathbf{v}' = [\mathbf{u}', 0, 0] \in \mathbb{R}^6$ represents a right isosceles triangle of orientation $\mathrm{sign}(A^*(\tilde{\mathbf{v}}')) = s\,\mathrm{sign}(\lambda)$ or colocated vertices.

The system $X\mathbf{u}' = \tilde{\mathbf{u}}'$ is solved by first finding the solutions of the homogeneous system $X\mathbf{u}_h = 0$ and translating them by a particular solution $\mathbf{u}_p$, obtaining $\mathbf{u}' = \mathbf{u}_h + \mathbf{u}_p$. The homogeneous system has an infinite number of solutions which come from the null space of X. This can be represented as a two-dimensional linear subspace $\mathbf{u}_h = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2$ where the coefficients $\beta_1, \beta_2 \in \mathbb{R}$ and with bases $\mathbf{u}_1, \mathbf{u}_2 \in$

$\mathbb{R}^4$ such that:

$$\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2\end{bmatrix} = \begin{bmatrix} 0 & -s\,\mathrm{sign}(\lambda) \\ s\,\mathrm{sign}(\lambda) & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{E.22}$$

We have that $\mathbf{u}_1$ and $\mathbf{u}_2$ represent two pairs of vertices which form right isosceles triangles when adding one third vertex in the origin of the same area $\frac{1}{4}$. The linear combination of $\mathbf{u}_h = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2$ generate right isosceles triangles of any area when adding one third vertex in the origin of orientation $-s\,\mathrm{sign}(\lambda)$. We then calculate the particular solution $\mathbf{u}_p$ using the pseudo-inverse as:

$$\mathbf{u}_p = X^\dagger\tilde{\mathbf{u}}' = \frac{1}{4}\begin{bmatrix} \tilde{x}'_a + \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\tilde{y}'_b \\ \tilde{y}'_a - \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\tilde{x}'_b \\ \tilde{x}'_b - \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\tilde{y}'_a \\ \tilde{y}'_b + \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\tilde{x}'_a \end{bmatrix} \tag{E.23}$$

We then translate the null space with the particular solution and obtain $\mathbf{u}' = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2 + \mathbf{u}_p$. This linear combination represents pairs of vectors. A noticeable property is stated in the following Lemma, whose proof is given in Appendix E.1.4.

Lemma 14. $\mathbf{u}_p = 0 \Rightarrow \mathbf{v}' = [\mathbf{u}', 0, 0]$ *is a right isosceles triangle. The converse is not true.*

The next step is to constrain these to the prescribed area and orientation. We have $\mathbf{v}' = [\mathbf{u}', 0, 0] \in \mathbb{R}^6$. After some minor algebraic manipulations, we obtain the signed area as:

$$A^*(\mathbf{v}') = (1 + s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))\,\mathrm{sign}(\lambda))\,\frac{A^*(\tilde{\mathbf{v}}')}{4} - s\,\mathrm{sign}(\lambda)\frac{\beta_1^2 + \beta_2^2}{2}. \tag{E.24}$$

When $A^*(\tilde{\mathbf{v}}') \neq 0$ we have $s\,\mathrm{sign}(\lambda) = \mathrm{sign}(A^*(\tilde{\mathbf{v}}'))$ thus $\mathrm{sign}(\lambda) = s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))$. However, when $A^*(\tilde{\mathbf{v}}') = 0$ we have $\mathrm{sign}(A^*(\mathbf{v}')) = -s\,\mathrm{sign}(\lambda)$, which implies that $\mathrm{sign}(\lambda) = -s$. Using the orientation constraint (7), we can express $\mathbf{u}'$ as:

$$\begin{aligned} & \mathbf{u}' = \beta_1\mathbf{u}_1 + \beta_2\mathbf{u}_2 + \mathbf{u}_p \\ \text{s.t.} \qquad & (s + \mathrm{sign}(\lambda)\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}')))\,\frac{A^*(\tilde{\mathbf{v}}')}{4} - \mathrm{sign}(\lambda)\frac{(\beta_1^2 + \beta_2^2)}{2} - A_o = 0 \end{aligned} \tag{E.25}$$

Because of the area and orientation constraints, and because $\mathbf{u}_1$ and $\mathbf{u}_2$ are rotated copies of each other, the family defined by equation (E.22) can be generated by simply rotating $\mathbf{u}_1$ by

$$\rho = \sqrt{\beta_1^2 + \beta_2^2} = \sqrt{\frac{\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))A^*(\tilde{\mathbf{v}}') - 4kA_o}{2}} \tag{E.26}$$

where k depends on the type of input:

$$k = \begin{cases} s \operatorname{sign}(A^*(\tilde{\mathbf{v}}')) & \text{if } A^*(\tilde{\mathbf{v}}) \neq 0 \\ -1 & \text{if } A^*(\tilde{\mathbf{v}}) = 0 \end{cases} \tag{E.27}$$

so that the area constraint is met and then rotate $\mathbf{u}_1$ by some arbitrary angle θ. We note that when $k = s \operatorname{sign}(A^*(\tilde{\mathbf{v}}')) = 1$, then $\rho \in \mathbb{R}$ as long as $A(\tilde{\mathbf{v}})/4 \geq A_o$ (which corresponds to setting F_3). We define a new basis vector $\mathbf{u}_c$ as:

$$\mathbf{u}_c = \rho \begin{bmatrix} 0 & ks & 1 & 0 \end{bmatrix}. \tag{E.28}$$

Finally the triangle vertices are translated to the original coordinates by adding the fixed vertex v_c, we obtain:

$$\mathbf{v} = \begin{bmatrix} \mathscr{R}(\theta)\mathbf{u}_c + \mathbf{u}_p & 0 & 0 \end{bmatrix}^\top + \begin{bmatrix} v_c & v_c & v_c \end{bmatrix}^\top \tag{E.29}$$

where $\mathscr{R}(\theta)$ is a block diagonal matrix replicating the 2D rotation matrix $R(\theta)$ two times as $\mathscr{R} = \operatorname{diag}(R(\theta), R(\theta))$. Equation (E.29) generates an infinite number of solutions with the same cost, as shown by the following Lemma, proved in Appendix E.1.4.

Lemma 15. *All solutions generated by equation (E.29) have the same cost.*

Appendix E.1.4. Proof of Lemmas

Proof of Lemma 9. We start with the forward implication: $F_1 \Rightarrow A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = 4$. In F_1, $\tilde{\mathbf{v}}$ represents a single point. This implies $A(\tilde{\mathbf{v}}) = 0$ and $D_o(\tilde{\mathbf{v}}) = 0$. Replacing these values in the depressed quartic equation (E.7) causes the coefficients p and r to become constants, and coefficient q to vanish (also the orientation constraint vanishes). The depressed quartic thus transforms into a bi-quadratic:

$$\lambda^4 - 32\lambda^2 + 256 = 0, \tag{E.30}$$

whose solutions are:

$$|\lambda| = 4. \tag{E.31}$$

We now turn to the reverse implication: $F_1 \Leftarrow A(\tilde{\mathbf{v}}) = 0$ and $|\lambda| = 4$. We substitute $|\lambda| = 4$ in equation (E.13), giving:

$$4s \operatorname{sign}(A^*(\tilde{\mathbf{v}})) \operatorname{sign}(\lambda) |A^*(\tilde{\mathbf{v}})| - D_o(\tilde{\mathbf{v}}) = 0. \tag{E.32}$$

Since $A(\tilde{\mathbf{v}}) = 0$, then the only solution that satisfies equation (E.32) for any given value of $s \operatorname{sign}(A^*(\tilde{\mathbf{v}})) \operatorname{sign}(\lambda)$ is with $D_o(\tilde{\mathbf{v}}) = 0$, which implies that the input triangle is collapsed into a single point, hence to F_1. □

Proof of Lemma 10. We start with the forward implication: $F_2 \Rightarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -4$. In F_2, $\tilde{\mathbf{v}}$ represents a right isosceles triangle and orientation inversion. This implies $A(\tilde{\mathbf{v}}) \neq 0$ and $\text{sign}(A^*(\tilde{\mathbf{v}})) = -s$. We perform a similarity transformation to $\tilde{\mathbf{v}}$ to bring the fixed vertex to the origin and another to the x-axis, scaled to normalise their distance, giving $\tilde{\mathbf{v}}' = [1, 0, 0, \text{sign}(A^*(\tilde{\mathbf{v}})), 0, 0]^\top$ where $\text{sign}(A^*(\tilde{\mathbf{v}}))$ determines the orientation of the triangle. We obtain $A^*(\tilde{\mathbf{v}}') = \frac{\text{sign}(A^*(\tilde{\mathbf{v}}))}{2}$ and $D_o(\tilde{\mathbf{v}}') = 2$, thus the coefficients of the depressed quartic equation (E.15) become:

$$p = -\frac{32A_o - 8}{A_o} \tag{E.33}$$

$$q = \frac{64}{A_o} \tag{E.34}$$

$$r = \frac{256A_o + 128}{A_o}. \tag{E.35}$$

Substituting these coefficients in Cardano's formula we obtain:

$$Q_1 = -\left(\frac{32A_o + 4}{3A_0}\right)^2 \tag{E.36}$$

$$Q_2 = \left(\frac{32A_o + 4}{3A_0}\right)^3, \tag{E.37}$$

making $\sqrt{Q_1^3 + Q_2^2} = 0$ and the real root α_o of Cardano's resolvent cubic to become:

$$\alpha_o = 2\left(\frac{32A_o + 4}{3A_o}\right) + \frac{32A_o - 8}{3A_o} = 32. \tag{E.38}$$

We finally use α_o to extract the roots of the depressed quartic:

$$\lambda = \frac{s_1\sqrt{32}\lambda_o + s_2\sqrt{8\left(\frac{1-s_1}{A_o}\right)}}{\sqrt{2}} \tag{E.39}$$

where $s_1, s_2 \in \{-1, 1\}$. We thus have the following roots:

$$\lambda \in \left\{-4, -4, 4 - 2i\sqrt{\frac{2}{A_o}}, 4 + 2i\sqrt{\frac{2}{A_o}}\right\}. \tag{E.40}$$

Considering only the real roots we have $\lambda = -4$.

We now turn to the reverse implication: $F_2 \Leftarrow A(\tilde{\mathbf{v}}) \neq 0$ and $\lambda = -4$. We substitute $\lambda = -4$ in equation (E.13), giving:

$$-4s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))A(\tilde{\mathbf{v}}) - D_o(\tilde{\mathbf{v}}) = 0. \tag{E.41}$$

Since $A(\tilde{\mathbf{v}}) \neq 0$ and $D_o(\tilde{\mathbf{v}}) > 0$ then equation (E.41) can only be solved when $\mathrm{sign}(A^*(\tilde{\mathbf{v}})) = -s$. We perform the same similarity transformation used at the beginning of the proof to the unknown input triangle $\tilde{\mathbf{v}}$ giving $\tilde{\mathbf{v}}' = [1, 0, x_b', y_b', 0, 0]^\top$ where one of the vertices remains unknown. We then have $A^*(\tilde{\mathbf{v}}') = \frac{\tilde{y}_b'}{2}$ or $\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))A(\tilde{\mathbf{v}}') = \frac{\tilde{y}_b'}{2}$and $D_o(\tilde{\mathbf{v}}') = x_b'^2 + y_b'^2 + 1$. which we substitute in equation (E.41) and obtain:

$$x_b'^2 + y_b'^2 + 2sy_b' + 1 = 0, \tag{E.42}$$

which we rewrite as:

$$x_b'^2 + (y_b' + s)^2 = 0. \tag{E.43}$$

This is the equation of a single point, making $\tilde{\mathbf{v}}'$ a right isosceles triangle $\tilde{\mathbf{v}}' = [0, 1, 0, -s, 0, 0]^\top$ where s determines the orientation of the triangle. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\tilde{\mathbf{v}}$ is also a right isosceles triangle when $\mathrm{sign}(A^*(\tilde{\mathbf{v}})) = -s$, which corresponds to F_2. □

Proof of Lemma 11. In F_3, $\tilde{\mathbf{v}}$ represents a right isosceles triangle with $A(\tilde{\mathbf{v}})/4 \geq A_o$ and no orientation inversion. This implies $A^*(\tilde{\mathbf{v}}) \neq 0$ and $\mathrm{sign}(A^*(\tilde{\mathbf{v}})) = s$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 10, thus the coefficients of the depressed quartic equation (30) become:

$$p = -\frac{32A_o + 8}{A_o} \tag{E.44}$$

$$q = \frac{32}{A_o} \tag{E.45}$$

$$r = \frac{256A_o - 128}{A_o}. \tag{E.46}$$

Substituting these coefficients in Cardano's formula we obtain:

$$Q_1 = -\left(\frac{32A_o - 4}{3A_0}\right)^2 \tag{E.47}$$

$$Q_2 = -\left(\frac{32A_o - 4}{3A_0}\right)^3, \tag{E.48}$$

making $\sqrt{Q_1^3 + Q_2^2} = 0$. After some factoring we obtain the real root α_o of Cardano's resolvent cubic as:

$$\alpha_o = -2\sqrt[3]{\left(\frac{32A_o - 8sA^*(\tilde{\mathbf{v}}')}{3A_o}\right)^3} + \frac{32A_o + 16sA^*(\tilde{\mathbf{v}}')}{3A_o}. \tag{E.49}$$

Since $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$ and $A^*(\tilde{\mathbf{v}}') = \operatorname{sign}(A^*(\tilde{\mathbf{v}}))A(\tilde{\mathbf{v}}')$, then $\alpha_o \in \mathbb{R}$ only when $A(\tilde{\mathbf{v}}')/4 \geq A_o$. Under this condition, we obtain $\alpha_o = 32$. Substituting in equation (37) we obtain:

$$\lambda = \frac{s_1\sqrt{32}\lambda_o + s_2\sqrt{8\left(\frac{1-s_1}{A_o}\right)}}{\sqrt{2}} \tag{E.50}$$

where $s_1, s_2 \in \{-1, 1\}$. We thus have the following roots:

$$\lambda \in \left\{-4 - 2\sqrt{\frac{2}{A_o}}, -4 + 2\sqrt{\frac{2}{A_o}}, 4, 4\right\}^\top. \tag{E.51}$$

Thus, we have two roots where $|\lambda| \neq 4$ (which correspond to solutions for Case I) and there exists at least one solution $\lambda = 4$ for F_3 (which correspond to the solution of Case II). □

Proof of Lemma 12. We substitute $\lambda = 4$ in equation (E.13), giving:

$$4s\operatorname{sign}(A^*(\tilde{\mathbf{v}}))A(\tilde{\mathbf{v}}) - \sigma^2(\tilde{\mathbf{v}}) = 0. \tag{E.52}$$

Since $A(\tilde{\mathbf{v}}) \neq 0$ then equation (E.52) can only be solved when $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 10, substitute $\operatorname{sign}(A^*(\tilde{\mathbf{v}}'))A(\tilde{\mathbf{v}}')$ and $\sigma^2(\tilde{\mathbf{v}}')$ in equation (E.52) and obtain:

$$x_b'^2 + y_b'^2 - 2sy_b' + 1 = 0, \tag{E.53}$$

which we rewrite as:

$$x_b'^2 + (y_b' - s)^2 = 0. \tag{E.54}$$

This is the equation of a single point making $\tilde{\mathbf{v}}'$ a right isosceles triangle $\tilde{\mathbf{v}}' = [0, 1, 0, s, 0, 0]^\top$ where s determines the orientation of the triangle. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\tilde{\mathbf{v}}$ is also a right isosceles triangle when $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$ for any value $A(\tilde{\mathbf{v}})$ (including $A(\tilde{\mathbf{v}}')/4 \geq A_o$) which corresponds to F_3. □

Proof of Lemma 13. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 8 and obtain $A^*(\tilde{\mathbf{v}}') = \frac{\operatorname{sign}(A^*(\tilde{\mathbf{v}}))}{2}$. We relax the condition of F_3 where $\frac{A(\tilde{\mathbf{v}}')}{4} \geq A_o$ by reformulating $A_o = \frac{1}{2z}$ where z is a scaling factor $z > 0 \in \mathbb{R}$. We substitute this in equation (E.51) and extract the roots :

$$\lambda = \left[-4 - 4\sqrt{z}, -4 + 4\sqrt{z}, 4, 4\right]^\top . \tag{E.55}$$

We start with the solution given by Case I where $|\lambda| \neq 4$. We compact both values as $\lambda = -4 + s_3 4\sqrt{z}$ where $s_3 \in \{-1, 1\}$. We substitute this in equation (E.6) and obtain:

$$\mathbf{u}' = \begin{bmatrix} \frac{s_3}{\sqrt{z}} & 0 & 0 & \frac{s_3}{\sqrt{z}} \end{bmatrix}^\top . \tag{E.56}$$

We calculate the cost of the solution of Case I by substituting $\mathbf{v}' = [\mathbf{u}', \tilde{x}_c, \tilde{y}_c]$ and $\tilde{\mathbf{v}}'$ in equation (3) and obtain:

$$\mathscr{C}_1(\mathbf{v}') = \frac{2(\sqrt{z} - s_3)^2}{z} \tag{E.57}$$

Now we turn to the solution given by Case II where $\lambda = 4$. We calculate the basis vector $\mathbf{u}'_c$, the particular solution $\mathbf{u}'_p$ and substitute them in equation (E.29) and obtain:

$$\mathbf{v}' = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{z-4}}{2\sqrt{z}} & \frac{\sqrt{z-4}}{2\sqrt{z}} & \frac{1}{2} & 0 & 0 \end{bmatrix}^\top . \tag{E.58}$$

We calculate the cost of the solution of Case II by substituting this and $\tilde{\mathbf{v}}'$ in equation (3) and obtain:

$$\mathscr{C}_2(\mathbf{v}') = 1 - \frac{2}{z} \tag{E.59}$$

We compare the cost of both solutions $\mathscr{C}_1(\mathbf{v}') \geq \mathscr{C}_2(\mathbf{v}')$ and obtain:

$$\frac{2(\sqrt{z} - s_3)^2}{z} \geq 1 - \frac{2}{z} \tag{E.60}$$

After some minor manipulations we obtain:

$$\frac{z - 4s_3\sqrt{z} + 4}{2z} \geq 0 \tag{E.61}$$

We substitute $\sqrt{z} = a$ where $a > 0 \in \mathbb{R}$ and obtain the quadratic expression:

$$\frac{(a - 4s_3)^2}{2a^2} \geq 0 \tag{E.62}$$

which represents an upward opening parabola which is always positive for any value of a and thus z. This implies that $\mathscr{C}_1(\mathbf{v}') \geq \mathscr{C}_2(\mathbf{v}')$ for any value z including $z > 4$. Since $\tilde{\mathbf{v}}'$ was a similarity transformation of $\tilde{\mathbf{v}}$, then $\mathscr{C}_1(\mathbf{v}) \geq \mathscr{C}_2(\mathbf{v})$. This means that in F_3 the solution provided by case II has the lowest cost, thus is the optimal solution. □

Proof of Lemma 14. We start with the forward implication: $\mathbf{u}_p = 0 \Rightarrow \mathbf{u}'$ is a pair of perpendicular vectors of equal length. From equation (E.23), we have that $\mathbf{u}_p = 0$ when $\tilde{\mathbf{u}}' = 0$ (trivial solution) or when $\tilde{\mathbf{u}}' = 0$ is in the kernel of $X^\dagger$. From the properties of the pseudo-inverse we have that $\ker(X^\dagger) = \ker(X^\top)$. Since X is symmetric then $\ker(X^\dagger) = \ker(X) = \mathbf{u}_h$, as given in equation (E.22). Recall from Appendix E.1.3 that the system $X\mathbf{u}' = \tilde{\mathbf{u}}'$ is solvable if and only if $\tilde{\mathbf{v}}' = [\tilde{\mathbf{u}}', 0, 0]$ represents a right isosceles triangle of orientation $\operatorname{sign}(A^*(\tilde{\mathbf{v}}')) = s \operatorname{sign}(\lambda)$ or colocated vertices. Still from Appendix E.1.3, we have that $\mathbf{v}_h = [\mathbf{u}_h, 0, 0]$ is a right isosceles triangle of orientation $\operatorname{sign}(A^*(\mathbf{v}_h)) = -s \operatorname{sign}(\lambda)$, which contradicts the previous statement. This leaves us with $\mathbf{u}_p = 0$ only when $\tilde{\mathbf{u}}' = 0$, which represents a set of colocated points centred in the origin. We then calculate $\mathbf{u}'$ from equation (E.25) and obtain:

$$\mathbf{u}' = \begin{bmatrix} -\beta_2 \\ \beta_1 \\ \beta_1 \\ \beta_2 \end{bmatrix}. \tag{E.63}$$

We can then easily show that the vertices distances with the origin are equal, and thus that $\mathbf{v}'$ is an isosceles triangle:

$$\begin{aligned} D_{ab} &= \sqrt{\beta_1^2 + \beta_2^2} \\ D_{ac} &= \sqrt{\beta_1^2 + \beta_2^2}. \end{aligned}$$

Also we can show that $\mathbf{v}'$ is a right isosceles triangle by calculating the inner product of its vertices:

$$I_{ab} = (x'_a \times x'_b) + (y'_a \times y'_b) = 0. \tag{E.64}$$

We now turn to the reverse implication: $\mathbf{u}_p = 0 \not\Leftarrow \mathbf{v}' = [\mathbf{v}', 0, 0]$ is a right isosceles triangle, for which we simply provide a counterexample to the positive implication. We use the right isosceles triangle $\tilde{\mathbf{v}}' = [1, 0, 0, \operatorname{sign}(A^*(\tilde{\mathbf{v}}), 0, 0]^\top$ where $A(\tilde{\mathbf{v}}')/4 = A_o$ and $\operatorname{sign}(A^*(\tilde{\mathbf{v}})) = s$. We calculate its particular solution with equation (E.23) and obtain:

$$\mathbf{u}_p = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix}, \tag{E.65}$$

which implies that $\mathbf{u}_p \neq 0$. We substitute $A(\tilde{\mathbf{v}}')/4 = A_o$ and $\text{sign}(A^*(\tilde{\mathbf{v}})) = s$ in the constraint equation (E.25) and obtain $\beta_1^2 = -\beta_2^2$. This implies that the constraint is fulfilled when $\beta_1 = \beta_2 = 0$ thus $\mathbf{u}' = \mathbf{u}_p$. We easily show that the inter-vertex distances are equal:

$$\begin{aligned} D_{ab} =& \sqrt{(x'_a - x'_b)^2 + (y'_a - y'_b)^2} = \frac{1}{2} \\ D_{ac} =& \sqrt{(x'_b - x'_c)^2 + (y'_b - y'_c)^2} = \frac{1}{2}, \end{aligned}$$

and also that the vertices are perpendicular by calculating their inner product:

$$I_{ab} = (x'_a \times x'_b) + (y'_a \times y'_b) = 0. \tag{E.66}$$

Thus, $\mathbf{v}$ is a right isosceles triangle despite $\mathbf{u}_p \neq 0$. □

Proof of Lemma 15. Our proof shows that the cost is invariant to the value chosen for angle θ. We start by translating the coordinate system to bring the fixed vertex to the origin as $\tilde{\mathbf{v}}' = \tilde{\mathbf{v}} - v_c$, which also translates the unknown vertices to $\mathbf{v}' = \mathbf{v} - v_c$. Thus, the cost becomes the translated least-squares displacement cost:

$$\mathscr{C}'(\mathbf{v}') = \|\mathbf{v}' - \tilde{\mathbf{v}}'\|^2. \tag{E.67}$$

We then continue with the case where $\tilde{\mathbf{v}}$ is a single point. This implies $A(\tilde{\mathbf{v}}) = 0$ and $k = -1$. The particular solution $\mathbf{v}'_p$ can be verified to vanish, from equation (E.23). We calculate the basis vector $\mathbf{v}'_c$ from equation (E.28), leading, from equation (E.29), to:

$$\mathbf{u}' = \begin{bmatrix} -\rho\sin(\theta) \\ \rho\cos(\theta) \\ \rho\cos(\theta) \\ \rho\sin(\theta) \end{bmatrix}. \tag{E.68}$$

We calculate the cost by substituting $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}'$ in equation (E.67):

$$\mathscr{C}'(\mathbf{v}') = 2\rho^2(\sin^2(\theta) + \cos^2(\theta)), \tag{E.69}$$

which simplifies to $\mathscr{C}'(\mathbf{v}') = 2\rho^2$ and is thus independent of θ.

We continue with the case where $\tilde{\mathbf{v}}$ is a right isosceles triangle. This implies that $A(\tilde{\mathbf{v}}) \neq 0$ and $k = s\,\mathrm{sign}(A^*(\tilde{\mathbf{v}}))$. We perform the same similarity transformation to $\tilde{\mathbf{v}}$ as in Lemma 10 and obtain $A^*(\tilde{\mathbf{v}}') = \frac{\mathrm{sign}(A^*(\tilde{\mathbf{v}}))}{2}$. We calculate the basis vector $\mathbf{u}_c'$ from equation (E.28), the particular solution $\mathbf{u}_p'$ from equation (E.23) and substitute them in equation (E.29), leading to:

$$\mathbf{u}' = \begin{bmatrix} \frac{1}{2} - \rho\sin(\theta) \\ \rho\cos(\theta) \\ \rho\cos(\theta) \\ \frac{1}{2} + \rho\sin(\theta) \end{bmatrix}. \tag{E.70}$$

We calculate the cost by substituting $\mathbf{v}'$ and $\tilde{\mathbf{v}}'$ in equation (E.67):

$$\mathscr{C}'(\mathbf{v}') = \frac{1}{2} + 2\rho^2(\sin^2(\theta) + \cos^2(\theta)), \tag{E.71}$$

which simplifies to $\mathscr{C}'(\mathbf{v}') = \frac{1}{2} + 2\rho^2$ and is thus independent of θ. □

Appendix E.1.5. Numerical Implementation

We use the theory developed in the previous sections to construct a numerically robust procedure, given in Algorithm 8, to solve OTPPAO with one fixed vertex. It uses the input vertices $\tilde{\mathbf{v}}$, prescribed area A_o and orientation s as inputs. It also uses an area error tolerance E to handle round-off in the area constraint (8). Similar to Algorithm 1, Algorithm 8 starts by generating the solutions from Case I, then Case II, and chooses the optimal one. For Case I, we obtain a list $\boldsymbol{v}_1$ of at most 4 solutions. For Case II, we obtain a single best solution $\mathbf{v}_2$, the optimally rotated one, the vertices basis and translation to generate all solutions following equation (E.29). The overall optimal solution $\mathbf{v}_o$ is chosen amongst $\boldsymbol{v}_1$ and $\mathbf{v}_2$. The algorithm returns the optimal solution, along with all the solutions from Case I and Case II.

Appendix E.2. Two Fixed Vertices

We assume that v_b and v_c are fixed. Thus, we redefine $\mathbf{v} = [v_a, \tilde{x}_b, \tilde{y}_b, \tilde{x}_c, \tilde{y}_c]^\top$ where v_a is the moving vertex. We take $\frac{\partial \mathscr{L}}{\partial v_a} = 0$ which are essentially the first two equalities of equation (12). We rearrange these equations into a set of linear equations:

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} \tilde{x}_a - \frac{s\lambda}{4}(\tilde{y}_c - \tilde{y}_b) \\ \tilde{y}_a - \frac{s\lambda}{4}(\tilde{x}_b - \tilde{x}_c) \end{bmatrix}. \tag{E.72}$$

We thus solve system (E.72) with two cases. In Case I, which is the most general, we have $v_b \neq v_c$. In Case II, we have $v_b = v_c$. The solution for the latter case is

Algorithm 8 Optimal Triangle Projection with a Prescribed Area and Orientation with a Fixed Vertex

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\mathbf{v}_o$ - optimal triangle, $\mathcal{v}_1$ - Case I triangle set, $\mathbf{v}_2$ - Case II optimal triangle, $\mathbf{u}_c, \mathbf{v}_t$ - Case II basis and offset
function OTTPAO1($\tilde{\mathbf{v}}, A_o, s, E = 10^{-3}$)
    $\mathcal{v}_1 \leftarrow$ SolveCase1OneFixedVertex($\tilde{\mathbf{v}}, A_o, s, E$)    ▷ Case I solutions
    $(\mathbf{v}_2, \mathbf{v}_c, \mathbf{u}_c, \mathbf{v}_t) \leftarrow$ SolveCase2OneFixedVertex($\tilde{\mathbf{v}}, A_o, s, E$)    ▷ Case II solutions
    $\mathbf{v}_o \leftarrow$ FindTriangleOfMinimalCost($\mathcal{v}_1 \cup \{\mathbf{v}_2\}$)    ▷ Optimal solution
    return $\mathbf{v}_o, \mathcal{v}_1, \mathbf{v}_2, \mathbf{u}_c, \mathbf{v}_t$
end function
```

trivial, since no matter what value we give to λ, the non-fixed vertex remains the same ($x_a = \tilde{x}_a$ and $y_a = \tilde{y}_a$). For Case I, we substitute the result of equation (E.72) in $\mathbf{v}$ and calculate the signed area constraint (8). The resulting linear equation in λ is $a_1\lambda + a_0 = 0$ where:

$$\begin{aligned} a_0 &= 4s\left((\tilde{x}_a - \tilde{x}_c)(\tilde{y}_b - \tilde{y}_a) - (\tilde{x}_a - \tilde{x}_b)(\tilde{y}_c - \tilde{y}_a)\right) - 8A_o \\ a_1 &= -(\tilde{x}_b^2 + \tilde{x}_c^2 + \tilde{y}_b^2 + \tilde{y}_c^2 - 2\tilde{x}_b\tilde{x}_c - 2\tilde{y}_b\tilde{y}_c). \end{aligned}$$

We can rewrite these coefficients more compactly. Concretely, a_0 contain the area $A^*(\tilde{\mathbf{v}})$ of the input vertices and a_1 contains the square distance P_o between the two fixed vertices:

$$P_o = (\tilde{x}_b - \tilde{x}_c)^2 + (\tilde{y}_b - \tilde{y}_c)^2. \tag{E.73}$$

We solve for λ and obtain:

$$\lambda = 8\frac{sA^*(\tilde{\mathbf{v}}) - A_o}{P_o}. \tag{E.74}$$

We substitute λ from equation (E.74) in equation (E.72) and obtain:

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} \tilde{x}_a \\ \tilde{y}_a \end{bmatrix} - 4\frac{sA_o + A^*(\tilde{\mathbf{v}})}{P_o}\begin{bmatrix} (\tilde{y}_c - \tilde{y}_b) \\ (\tilde{x}_b - \tilde{x}_c) \end{bmatrix}. \tag{E.75}$$

The numerically robust procedure of this solution is given in Algorithm 11.

Algorithm 9 Closed-form Analytic Solution to Case I of OTPPAO with One Fixed Vertex

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\boldsymbol{v}_1$ - solution list
function SolveCase1OneFixedVertex($\tilde{\mathbf{v}}, A_o, s, E$)
    $D_o(\tilde{\mathbf{v}})) \leftarrow (\tilde{x}_a - \tilde{x}_c)^2 + (\tilde{y}_a - \tilde{y}_c)^2 + (\tilde{x}_b - \tilde{x}_c)^2 + (\tilde{y}_b - \tilde{y}_c)^2$
    $p \leftarrow -\frac{16(2A_o + sA^*(\tilde{\mathbf{v}}))}{A_o}$    ▷ Compute the coefficients of the depressed quartic
    $q \leftarrow \frac{32D_o(\tilde{\mathbf{v}})}{A_o}$
    $r \leftarrow \frac{256(A_o - sA^*(\tilde{\mathbf{v}}))}{A_o}$
    $\boldsymbol{\lambda} \leftarrow$ FerrariSolution($p, q, r$)    ▷ Solve for the four possible Lagrange multipliers
    $\boldsymbol{v}_1 \leftarrow \emptyset$    ▷ Create an empty set of solutions
    for $t \leftarrow 1, \ldots, 4$ do    ▷ Generate and select the triangles
        $\lambda \leftarrow \mathrm{Re}(\boldsymbol{\lambda}(t))$    ▷ Keep the real part
        $h \leftarrow (\lambda^2 - 16)^\dagger$    ▷ Compute the inverse denominator
        $\mathbf{u} \leftarrow h \begin{bmatrix} \tilde{x}_c\lambda^2 + 4s(\tilde{y}_b - \tilde{y}_c)\lambda - 16\tilde{x}_a \\ \tilde{y}_c\lambda^2 + 4s(\tilde{x}_c - \tilde{x}_b)\lambda - 16\tilde{y}_a \\ \tilde{x}_c\lambda^2 + 4s(\tilde{y}_c - \tilde{y}_a)\lambda - 16\tilde{x}_b \\ \tilde{y}_c\lambda^2 + 4s(\tilde{x}_a - \tilde{x}_c)\lambda - 16\tilde{y}_b \end{bmatrix}$    ▷ Compute the vertices
        $\mathbf{v} = [\mathbf{u}, \tilde{x}_c, \tilde{y}_c]$
        if $|sA^*(\mathbf{v}) - A_o| \leq E$ then    ▷ Check the area constraint
            $\boldsymbol{v}_1 \leftarrow \boldsymbol{v}_1 \cup \{\mathbf{v}\}$    ▷ Add the vertices to the solution set
        end if
    end for
    return $\boldsymbol{v}_1$
end function
```

Algorithm 10 Closed-form Analytic Solution to Case II of OTPPAO with One Fixed Vertex

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - area error tolerance
Output: $\mathbf{v}_2$ - optimal triangle, $\mathbf{u}_c, \mathbf{v}_t$ vertices basis and translation.
function SolveCase2OneFixVert($\tilde{\mathbf{v}}, A_o, s, E$)
    if $A(\mathbf{v}) \leq E$ then                                          ▷ Check the input's area
        $k \leftarrow s\,\mathrm{sign}(A^*(\mathbf{v}))$                      ▷ Compute $k$ for a right isosceles triangle
    else
        $k \leftarrow -1$                                                   ▷ Compute $k$ for a single point
    end if
    $\tilde{\mathbf{v}}' \leftarrow \tilde{\mathbf{v}} - v_c$                    ▷ Translate the input vertices
    $\rho \leftarrow \sqrt{\frac{\mathrm{sign}(A^*(\tilde{\mathbf{v}}'))A^*(\tilde{\mathbf{v}}')-4kA_o}{2}}$   ▷ Computes the area constraint parameter
    $\mathbf{u}_c \leftarrow \mathrm{Re}(\rho)\begin{bmatrix}0 & ks & 1 & 0\end{bmatrix}^\top$     ▷ Compute the solution basis
    $\mathbf{u}_p \leftarrow \frac{1}{4}\begin{bmatrix}\tilde{x}'_a + \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\tilde{y}'_b \\ \tilde{y}'_a - \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\tilde{x}'_b \\ \tilde{x}'_b - \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\tilde{y}'_a \\ \tilde{y}'_b + \mathrm{sign}(A^*(\tilde{\mathbf{v}}))\tilde{x}'_a\end{bmatrix}$   ▷ Compute the particular solution
    $\tilde{U}_2 \leftarrow$ rearrange $\tilde{\mathbf{u}}'$ into a $2 \times 2$ matrix
    $U_c \leftarrow$ rearrange $\mathbf{u}_c$ into a $2 \times 2$ matrix
    $(U_1, \Sigma, U_2) \leftarrow \mathrm{SVD}\left(\tilde{U}_2 U_c^\top\right)$          ▷ Compute the optimal rotation
    $D \leftarrow \mathrm{diag}(1, \det(U_2 U_1^\top))$
    $R \leftarrow U_2 D U_1^\top$
    $\mathbf{v}_t \leftarrow \begin{bmatrix}\mathbf{u}_p & 0 & 0\end{bmatrix}^\top + \begin{bmatrix}v_c & v_c & v_c\end{bmatrix}^\top$   ▷ Compute the translation vector
    $\mathbf{v}_2 = \begin{bmatrix}\mathscr{R}(\theta)\mathbf{u}_c & 0 & 0\end{bmatrix}^\top + \mathbf{v}_t$      ▷ Compute the optimal solution
    return $\mathbf{v}_2, \mathbf{u}_c, \mathbf{v}_t$
end function
```

Algorithm 11 Optimal Triangle Projection with a Prescribed Area and Orientation with Two Fixed Vertices

```
Input: $\tilde{\mathbf{v}}$ - input vertices, $A_o$ - prescribed area, $s$ - prescribed orientation, $E$ - distance error tolerance
Output: $\mathbf{v}_o$ - optimal triangle
function OTTPAO2($\tilde{\mathbf{v}}, A_o, s, E = 10^{-3}$)
    $P_o = (\tilde{x}_b - \tilde{x}_c)^2 + (\tilde{y}_b - \tilde{y}_c)^2$    ▷ Compute square distance between fixed vertices
    if $P_o > E$ then
        $\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} \tilde{x}_a \\ \tilde{y}_a \end{bmatrix} - 4P_o^{\dagger}(sA_o + A^*(\tilde{\mathbf{v}})) \begin{bmatrix} (\tilde{y}_c - \tilde{y}_b) \\ (\tilde{x}_b - \tilde{x}_c) \end{bmatrix}$
    else
        $\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} \tilde{x}_a \\ \tilde{y}_a \end{bmatrix}$
    end if
    $\mathbf{v}_o \leftarrow \begin{bmatrix} x_a & y_a & \tilde{x}_b & \tilde{y}_b & \tilde{x}_c & \tilde{y}_c \end{bmatrix}^\top$    ▷ Compute the optimal solution
    return $\mathbf{v}_o$
end function
```

References

[1] S. Sun, M. Zhang, G. Zhihong, Smoothing Algorithm for Planar and Surface Mesh Based on Element Geometric Deformation, Mathematical Problems in Engineering 2015 (2015) 1–9. doi:10.1155/2015/435648.

[2] D. Vartziotis, T. Athanasiadis, I. Goudas, J. Wipper, Mesh smoothing using the Geometric Element Transformation Method, Computer Methods in Applied Mechanics and Engineering 197 (45) (2008) 3760–3767. doi:10.1016/j.cma.2008.02.028.

[3] G. Irving, C. Schroeder, R. Fedkiw, Volume conserving finite element simulations of deformable models, ACM Transactions on Graphics (TOG) 26 (3) (2007) 13–es. doi:10.1145/1276377.1276394.
URL `https://doi.org/10.1145/1276377.1276394`

[4] N. Abu Rumman, M. Fratarcangeli, Position-Based Skinning for Soft Articulated Characters, Computer Graphics Forum 34 (6) (2015) 240–250. doi:10.1111/cgf.12533.
URL `https://doi.org/10.1111/cgf.12533`

[5] T.-C. Tsai, Position Based Dynamics, in: N. Lee (Ed.), Encyclopedia of Computer Graphics and Games, Springer International Publishing, Cham, 2017, pp. 1–5. doi:10.1007/978-3-319-08234-9-92-1.
URL https://doi.org/10.1007/978-3-319-08234-9-92-1

[6] M. Müller, B. Heidelberger, M. Hennix, J. Ratcliff, Position Based Dynamics, J. Vis. Comun. Image Represent. 18 (2) (2007) 109–118. doi:10.1016/j.jvcir.2007.01.005.
URL http://dx.doi.org/10.1016/j.jvcir.2007.01.005

[7] J. Bender, M. Müller, M. A. Otaduy, M. Teschner, M. Macklin, A Survey on Position-Based Simulation Methods in Computer Graphics, Computer Graphics Forum 33 (6) (2014) 228–251. doi:10.1111/cgf.12346.
URL https://doi.org/10.1111/cgf.12346

[8] M. Kelager, S. Niebe, K. Erleben, A Triangle Bending Constraint Model for Position-Based Dynamics, The Eurographics Association, 2010. doi:http://dx.doi.org/10.2312/PE/vriphys/vriphys10/031-037.

[9] Y. Wang, X. Wu, G. Wang, An Angle Bending Constraint Model for Position-Based Dynamics, in: 2014 International Conference on Virtual Reality and Visualization, 2014, pp. 430–434, iSSN: null. doi:10.1109/ICVRV.2014.58.

[10] M. Tournier, M. Nesme, B. Gilles, F. Faure, Stable constrained dynamics, ACM Transactions on Graphics (TOG) 34 (4) (2015) 132:1–132:10. doi:10.1145/2766969.
URL https://doi.org/10.1145/2766969

[11] Y. Gu, K. Yang, C. Lu, Constraint Solving Order in Position Based Dynamics, in: Proceedings of the 2017 2nd International Conference on Electrical, Automation and Mechanical Engineering (EAME 2017), 2017. doi:10.2991/eame-17.2017.46.

[12] S. Bouaziz, S. Martin, T. Liu, L. Kavan, M. Pauly, Projective dynamics: Fusing constraint projections for fast simulation Article No. 154doi:10.1145/2601097.2601116.
URL https://repository.upenn.edu/hms/146

[13] J.-P. Tignol, Galois' Theory Of Algebraic Equations, Wspc, Singapore ; River Edge, NJ, 2001.

[14] G. Cardano, T. R. Witmer, O. Ore, The Rules of Algebra: Ars Magna, Courier Corporation, 1545, google-Books-ID: ZMZY79bvOPgC.

[15] D. Herbison-Evans, Solving Quartics and Cubics for Graphics, in: A. W. Paeth (Ed.), Graphics Gems V, Academic Press, Boston, 1995, pp. 3–15. doi:10.1016/B978-0-12-543457-7.50009-7.

[16] H. Helfgott, M. Helfgott, A Modern Vision of the Work of Cardano and Ferrari on Quartics, Convergence (Feb. 2010).

[17] S. L. Shmakov, A Universal Method of Solving Quartic Equations, International Journal of Pure and Applied Mathematics 71 (2) (2011) 251–259, publisher: Academic Publications, Ltd.
URL `https://ijpam.eu/contents/2011-71-2/7/index.html`

[18] K. Arun, T. Huang, S. Blostein, Least-squares fitting of two 3-D point sets, Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-9 (1987) 698 – 700. doi:10.1109/TPAMI.1987.4767965.

[19] P.-O. Persson, G. Strang, A Simple Mesh Generator in MATLAB, SIAM Review 46 (2) (2004) 329–345. doi:10.1137/S0036144503429121.
URL `https://epubs.siam.org/doi/abs/10.1137/S0036144503429121`

[20] M. Friendly, G. Monette, J. Fox, Elliptical Insights: Understanding Statistical Methods through Elliptical Geometry, Statistical Science 28 (Feb. 2013). doi:10.1214/12-STS402.