Exact and Efficient Intersection Resolution for Mesh Arrangements — Supplements

JIA-PENG GUO and XIAO-MING FU*, University of Science and Technology of China, China

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: mesh arrangements, intersection resolution, constrained triangulation, geometric predicates

ACM Reference Format:
Jia-Peng Guo and Xiao-Ming Fu. 2024. Exact and Efficient Intersection Resolution for Mesh Arrangements — Supplements. *ACM Trans. Graph.* 43, 6, Article 165 (December 2024), 2 pages. https://doi.org/10.1145/3687925

1 ARITHMETICS AND CODE GENERATION

In the main content of the paper, we introduce the mathematical expressions for implicit points and part of the indirect offset predicates. In this supplementary, we explain how to use arithmetic filtering techniques to quickly and exactly evaluate predicates and provide details on generating code for this evaluation.

We briefly review the process of evaluating indirect offset predicates that receive any combination of explicit and implicit points. We first evaluate the terms of the implicit points' expression, $\beta+\lambda/d$, and determine the sign of d. Next, using the explicit points' coordinates and the implicit points' evaluated values, we evaluate the predicate $\Lambda = \Delta/D$ and determine the sign of Δ. Here, D is the product of all d values for the implicit points, and its sign is determined by the previously determined signs of d.

We use a three-stage arithmetic filtering model to determine the signs of d and Δ in the above process: floating-point arithmetic (semi-static filtering), interval arithmetic (dynamic filtering), and floating-point expansion arithmetic (exact). In the semi-static filtering stage, a static numerical error analysis is applied to the expression of implicit points and predicates to obtain the static numerical error bounds (ε_0) for results such as λ, d, and Δ [Meyer 2008]. Briefly introducing the analysis, it assumes the value bound of each input (e.g., coordinates of explicit points and coordinates involved in the construction of implicit points) to be 1 and the numerical error of each input to be 0, and then propagating the value bounds and errors according to the computation process of the expression, resulting in the final numerical error bounds for results. Automated analysis tools are provided in previous works [Attene 2020; Lévy 2016; Meyer 2008] and in our open-source project, too. After obtaining the static error bound ε_0, semi-static filtering scales the ε_0 to the actual error bound ε based on the actual value bounds δ of inputs, typically resulting in an expression of the form $\varepsilon = \varepsilon_0\delta^n$. Finally, if the magnitude of the value calculated using floating-point arithmetic is greater than the actual error, the sign of the value is guaranteed to be accurate. Otherwise, the next stages are used to determine the sign.

In the latter two stages, interval arithmetic [Brönnimann et al. 1998; Hemmer et al. 2023] and floating-point expansion arithmetic [Lévy 2016] are used to compute the expression and reliably determine the sign of the result. The former computes the possible intervals within which the intermediate and final results lie and checks if the result interval's sign is unambiguous. The latter uses floating-point expansion to exactly represent all intermediate and final results, allowing for an accurate determination of the result's sign.

We convert the three-stage arithmetic filtering process for implicit points and predicates from mathematical expressions into C++ code. The mathematical expressions of implicit points are introduced in the main content of the paper, and the mathematical expressions of all used predicates are listed in Section 2 and 3. By expanding the mathematical expressions into a series of binary operations, we can directly convert them into floating-point arithmetic code. After the floating-point computation is completed, we evaluate the maximal magnitude of the inputs, scale the error bound obtained by the automated analysis tool to the actual error bound, and determine the sign by comparing the result value with the actual error bound. Similarly, by using interval number types and floating-point expansion types with overloaded operators, we can directly convert the expressions into interval arithmetic code and floating-point expansion arithmetic code, then determine the sign of the result after the computation is complete. We develop this conversion tool based on Attene's tool [Attene 2020] and provide it in our open-source project.

2 GENERALIZED ORIENT2D

We list the expression and semi-static filters for the generalized `orient2d` predicates with different combinations of implicit points in indirect offset predicates. We only display predicates on the orthogonal plane XY since the expression and filters are essentially the same across different planes. We group combinations of implicit points based on how many implicit points are contained in the combination. Combinations containing 1, 2, or 3 implicit points are abbreviated as "IEE", "IIE", and "III", respectively, where "I" represents implicit points and "E" represents explicit points. "I" can be replaced with specific types of implicit points (LLI as "S", LPI as "L", and TPI as "T") to represent specific implicit point combinations. Predicates with the same number of implicit points have the same expressions but different filters.

*The corresponding author

Authors' address: Jia-Peng Guo, gjp171499@mail.ustc.edu.cn; Xiao-Ming Fu, fuxm@ustc.edu.cn, University of Science and Technology of China, China.

$$\texttt{orient2d\_XY\_IEE}(p_{I1}, p_{E2}, p_{E3}) = \texttt{sign}(\Delta)\texttt{sign}(d_{I1})$$

$$\begin{aligned}
\Delta =& (d_{I1}(\beta_{I1x} - p_{E3x}) + \lambda_{I1x})(p_{E2y} - p_{E3y}) \\
&- (d_{I1}(\beta_{I1y} - p_{E3y}) + \lambda_{I1y})(p_{E2x} - p_{E3x}) \\
\delta_\Delta =& \max\{\delta_I, |\beta_{I1x} - p_{E3x}|, |\beta_{I1y} - p_{E3y}| \\
& |p_{E2x} - p_{E3x}|, |p_{E2y} - p_{E3y}|\} \\
\varepsilon_{\Delta SEE} =& 8.881784197001255\ 10^{-15}\delta_\Delta^4 \\
\varepsilon_{\Delta LEE} =& 3.197724203485299\ 10^{-14}\delta_\Delta^5 \\
\varepsilon_{\Delta TEE} =& 1.0409451078885486\ 10^{-12}\delta_\Delta^8
\end{aligned}$$

$$\texttt{orient2d\_XY\_IIE}(p_{I1}, p_{I2}, p_{E3}) = \texttt{sign}(\Delta)\texttt{sign}(d_{I1})\texttt{sign}(d_{I2})$$

$$\begin{aligned}
\Delta =& (d_{I1}(\beta_{I1x} - p_{E3x}) + \lambda_{I1x})(d_{I2}(\beta_{I2y} - p_{E3y}) + \lambda_{I2y}) \\
&- (d_{I1}(\beta_{I1y} - p_{E3y}) + \lambda_{I1y})(d_{I2}(\beta_{I2x} - p_{E3x}) + \lambda_{I2x}) \\
\delta_\Delta =& \max\{\delta_{I1}, \delta_{I2}, |\beta_{I1x} - p_{E3x}|, |\beta_{I1y} - p_{E3y}|, \\
& |\beta_{I2x} - p_{E3x}|, |\beta_{I2y} - p_{E3y}|\} \\
\varepsilon_{\Delta SSE} =& 5.684341886080809\ 10^{-14}\delta_\Delta^6 \\
\varepsilon_{\Delta LSE} =& 1.918578143578212\ 10^{-13}\delta_\Delta^7 \\
\varepsilon_{\Delta TSE} =& 5.314859663485564\ 10^{-12}\delta_\Delta^{10} \\
\varepsilon_{\Delta LLE} =& 6.750832531876594\ 10^{-13}\delta_\Delta^8 \\
\varepsilon_{\Delta TLE} =& 1.7707333863081813\ 10^{-11}\delta_\Delta^{11} \\
\varepsilon_{\Delta TTE} =& 4.189644187135872\ 10^{-10}\delta_\Delta^{14}
\end{aligned}$$

$$\texttt{orient2d\_XY\_III}(p_{I1}, p_{I2}, p_{I3}) = \texttt{sign}(\Delta)\texttt{sign}(d_{I1})\texttt{sign}(d_{I2})\texttt{sign}(d_{I3})$$

$$\begin{aligned}
\Delta =& (d_{I3}(d_{I1}(\beta_{I1x} - \beta_{I3x}) + \lambda_{I1x}) - d_{I1}\lambda_{I3x}) \\
& (d_{I3}(d_{I2}(\beta_{I2y} - \beta_{I3y}) + \lambda_{I2y}) - d_{I2}\lambda_{I3y}) \\
& - (d_{I3}(d_{I1}(\beta_{I1y} - \beta_{I3y}) + \lambda_{I1y}) - d_{I1}\lambda_{I3y}) \\
& (d_{I3}(d_{I2}(\beta_{I2x} - \beta_{I3x}) + \lambda_{I2x}) - d_{I2}\lambda_{I3x}) \\
\delta_\Delta =& \max\{\delta_{I1}, \delta_{I2}, \delta_{I3}, |\beta_{I1x} - \beta_{I3x}|, |\beta_{I1y} - \beta_{I3y}| \\
& |\beta_{I2x} - \beta_{I3x}|, |\beta_{I2y} - \beta_{I3y}|\} \\
\varepsilon_{\Delta SSS} =& 8.455458555545215\ 10^{-13}\delta_\Delta^{10} \\
\varepsilon_{\Delta LSS} =& 2.746382982143922\ 10^{-12}\delta_\Delta^{11} \\
\varepsilon_{\Delta TSS} =& 5.881872766622135\ 10^{-11}\delta_\Delta^{14} \\
\varepsilon_{\Delta LLS} =& 9.152602287176878\ 10^{-12}\delta_\Delta^{12} \\
\varepsilon_{\Delta TLS} =& 1.9107143645058585\ 10^{-10}\delta_\Delta^{15} \\
\varepsilon_{\Delta TTS} =& 3.771219780901469\ 10^{-9}\delta_\Delta^{18} \\
\varepsilon_{\Delta LLL} =& 9.823293567468065\ 10^{-11}\delta_\Delta^{14} \\
\varepsilon_{\Delta TLL} =& 1.968414466146938\ 10^{-9}\delta_\Delta^{17} \\
\varepsilon_{\Delta TTL} =& 3.784163138398379\ 10^{-8}\delta_\Delta^{20} \\
\varepsilon_{\Delta TTT} =& 1.1310143236187382\ 10^{-5}\delta_\Delta^{26}
\end{aligned}$$

3 POINT COMPARE

We list the expression and semi-static filters for the `pointCompare` predicate. The `longestAxis` predicate has similar expressions and semi-static filters with `pointCompare` predicate to prevent degeneration on the selected axis.

$$\texttt{pointCompare\_X\_IE}(p_{I1}, p_{E2}) = \texttt{sign}(\Delta)\texttt{sign}(d_{I1})$$

$$\begin{aligned}
\Delta =& d_{I1}(\beta_{I1x} - p_{E2x}) + \lambda_{I1x} \\
\delta_\Delta =& \max\{\delta_{I1}, |\beta_{I1x} - p_{E2x}|\} \\
\varepsilon_{\Delta SE} =& 3.108624468950439\ 10^{-15}\delta_\Delta^3 \\
\varepsilon_{\Delta LE} =& 1.2879996548476053\ 10^{-14}\delta_\Delta^4 \\
\varepsilon_{\Delta TE} =& 4.680700271819666\ 10^{-12}\delta_\Delta^7
\end{aligned}$$

$$\texttt{pointCompare\_X\_II}(p_{I1}, p_{I2}) = \texttt{sign}(\Delta)\texttt{sign}(d_{I1})\texttt{sign}(d_{I2})$$

$$\begin{aligned}
\Delta =& d_{I1}d_{I2}(\beta_{I1x} - \beta_{I2x}) + \lambda_{I1x}d_{I2} - \lambda_{I2x}d_{I1} \\
\delta_\Delta =& \max\{\delta_{I1}, \delta_{I2}, |\beta_{I1x} - \beta_{I2x}|\} \\
\varepsilon_{\Delta SS} =& 1.6431300764452333\ 10^{-14}\delta_\Delta^5 \\
\varepsilon_{\Delta LS} =& 6.084585960075558\ 10^{-14}\delta_\Delta^6 \\
\varepsilon_{\Delta TS} =& 1.6022738691390292\ 10^{-12}\delta_\Delta^9 \\
\varepsilon_{\Delta LL} =& 2.20746164403263\ 10^{-13}\delta_\Delta^7 \\
\varepsilon_{\Delta TL} =& 5.27253154330999\ 10^{-12}\delta_\Delta^{10} \\
\varepsilon_{\Delta TT} =& 1.0712852827055069\ 10^{-10}\delta_\Delta^{13}
\end{aligned}$$

REFERENCES

Marco Attene. 2020. Indirect Predicates for Geometric Constructions. *Comput. Aided Des.* 126 (2020), 102856.

Hervé Brönnimann, Christoph Burnikel, and Sylvain Pion. 1998. Interval Arithmetic Yields Efficient Dynamic Filters for Computational Geometry. In *Proc. Annu. Symp. Comput. Geom. (SCG '98)*. 165–174.

Michael Hemmer, Susan Hert, Sylvain Pion, and Stefan Schirra. 2023. Number Types. In *CGAL User and Reference Manual* (5.6 ed.). CGAL Editorial Board. https://doc.cgal.org/5.6/Manual/packages.html#PkgNumberTypes

Bruno Lévy. 2016. Robustness and efficiency of geometric programs: The Predicate Construction Kit (PCK). *Comput. Aided Des.* 72 (2016), 3–12.

Pion Meyer. 2008. FPG A code generator for fast and certified geometric predicates. *Real Numbers and Computers* (2008).