Compact and efficient implicit representations

Clément Laroche

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

Compact and efficient implicit representations

Clément Laroche

ATHENS

APRIL 2020

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χρονικά και χωρικά αποδοτικές αλγεβρικές αναπαραστάσεις

Clément Laroche

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2020

PhD THESIS

Compact and efficient implicit representations

Clément Laroche

SUPERVISOR: Ioannis Z. Emiris, Professor NKUA
THREE-MEMBER ADVISORY COMMITTEE:

Ioannis Z. Emiris, Professor NKUA

Missirlis Nikolaos, Emeritus Professor NKUA

Bernard Mourrain, Research Director INRIA

SEVEN-MEMBER EXAMINATION COMMITTEE

Ioannis Z. Emiris,
Professor NKUA

Missirlis Nikolaos,
Emeritus Professor NKUA

Bernard Mourrain,
Research Director INRIA

Aristides Kontogeorgis,
Professor NKUA

Michael N. Vrahatis,
Professor UP

Dimitrios Poulakis,
Professor AUTH

Ilias S. Kotsireas,
Professor WLU

Examination Date: 30th of April 2020

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χρονικά και χωρικά αποδοτικές αλγεβρικές αναπαραστάσεις

Clément Laroche

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Ζ. Εμίρης, Καθηγητής ΕΚΠΑ
ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Ιωάννης Ζ. Εμίρης, Καθηγητής ΕΚΠΑ

Μισυρλής Νικόλαος, Επίτιμος Καθηγητής ΕΚΠΑ

Bernard Mourrain, Διευθυντής Έρευνας INRIA

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Ιωάννης Ζ. Εμίρης,
Καθηγητής ΕΚΠΑ

Μισυρλής Νικόλαος,
Επίτιμος Καθηγητής ΕΚΠΑ

Bernard Mourrain,
Διευθυντής Έρευνας INRIA

Αριστείδης Κοντογεώργης,
Καθηγητής ΕΚΠΑ

Μιχαήλ Βραχάτης του Νικήτα,
Καθηγητής ΠΠ

Δημήτριος Πουλάκης,
Καθηγητής ΑΠΘ

Ηλίας Κοτσιρέας,
Καθηγητής WLU

Ημερομηνία Εξέτασης: 30 Απριλίου 2020

ABSTRACT

In the perspective of manipulating geometric objects, there exists two main representations of curves and surfaces: parametric and implicit representations. Both are useful for different purposes and thus complement each other. Parametric representations are efficient in sampling points on an object; implicit representations are efficient in determining whether a point belongs to an object or not. Because of that, having both representations of the same objects at the same time maximizes the range of operations one can do with geometric objects. Switching from one representation to another is not an easy task. It usually requires the use of algebraic properties. Thus, there is a strong link between algebra and geometry, symbolised by the algebraic varieties: they are geometric objects described by an algebraic structure.

This thesis explores new kinds of implicit representations and algorithms for computing implicit representations. We show that different methods are adapted to different situations even when it comes to the choice of an implicit representation amongst several possibilities. Space curves can thus be described implicitly by conical surfaces, moving lines and/or moving quadrics…each description having different geometrical properties and practical usage. As there is not one implicit representation or implicitization algorithm that would be the best in any situation, we develop methods that fit to different kinds of informations known about the object we want to represent. As we show, objects constructed by sweeping a rigid body can be represented using the knowledge of that nature. Similarly, very particular curves may have a complicated algebraic structure. Depending on our tolerance to approximation, such curves can thus be perturbed to simplify greatly their algebraic structure or, on the contrary, be represented by a rich implicit representation format.

SUBJECT AREA: Algebraic Geometry

KEYWORDS: Algebraic varieties, Implicitization, CAGD-CAE, Resultants, Syzygies

ΠΕΡΙΛΗΨΗ

Στην προοπτική του χειρισμού γεωμετρικών αντικειμένων, υπάρχουν δύο κύριες αναπαραστάσεις καμπυλών και επιφανειών: παραμετρικές και αλγεβρικές αναπαραστάσεις. Και οι δύο είναι χρήσιμες για διαφορετικούς σκοπούς και έτσι αλληλοσυμπληρώνονται. Οι παραμετρικές αναπαραστάσεις είναι αποτελεσματικές στα σημεία δειγματοληψίας ενός αντικειμένου. Οι αλγεβρικές αναπαραστάσεις είναι αποτελεσματικές για τον προσδιορισμό του αν ένα σημείο ανήκει σε ένα αντικείμενο ή όχι. Εξαιτίας αυτού, η κατοχή και των δύο αναπαραστάσεων για το ίδιο αντικείμενο μεγιστοποιεί το εύρος των λειτουργιών που μπορούν να πραγματοποιηθούν με το αντικείμενο αυτό. Η μετάβαση από μία αναπαράσταση σε άλλη δεν είναι εύκολη υπόθεση. Απαιτεί συνήθως τη χρήση αλγεβρικών ιδιοτήτων. Έτσι, υπάρχει μια ισχυρή σχέση μεταξύ άλγεβρας και γεωμετρίας, που συμβολίζεται από τα αλγεβρικά σύνολα: είναι γεωμετρικά αντικείμενα που περιγράφονται από αλγεβρικές εξισώσεις. Η μετάβαση από την παραμετρική στην αλγεβρική αναπαράσταση καλείται αλγεβρικοποίηση.

Αυτή η εργασία εξετάζει νέα είδη αλγεβρικών αναπαραστάσεων και αλγορίθμων για τον υπολογισμό αλγεβρικών αναπαραστάσεων. Δείχνουμε ότι διάφορες μέθοδοι προσαρμόζονται σε διαφορετικές καταστάσεις, ακόμη και όταν πρόκειται για την επιλογή μιας αλγεβρικής αναπαράστασης μεταξύ διαφόρων αλγεβρικών αναπαραστάσεων. Οι καμπύλες στον χώρο μπορούν έτσι να περιγραφούν αλγεβρικά με κωνικές επιφάνειες, κινούμενες γραμμές και/ή κινούμενη τετραγωνική επιφάνεια…όπου κάθε περιγραφή έχει διαφορετικές γεωμετρικές ιδιότητες και πρακτική χρήση. Εφόσον δεν υπάρχει βέλτιστη αναπαράσταση ή βέλτιστος αλγόριθμος αλλαγής αναπαράστασης για όλες τις περιπτώσεις, επιλέγουμε να αναπτύξουμε μεθόδους που ταιριάζουν σε διάφορα είδη γνωστών πληροφοριών σχετικά με το αντικείμενο που θέλουμε να αντιπροσωπεύουμε. Όπως δείχνουμε, τα αντικείμενα που κατασκευάζονται με τη μετατόπιση ενός άκαμπτου σώματος (swept volumes) μπορούν να αναπαρίστανται χρησιμοποιώντας τη γνώση αυτής της φύσης. Ομοίως, πολύ συγκεκριμένες καμπύλες μπορεί να έχουν περίπλοκη αλγεβρική δομή. Ανάλογα με την ανοχή μας στην προσέγγιση, τέτοιες καμπύλες μπορούν έτσι να διαταραχθούν για να απλουστεύσουν σε μεγάλο βαθμό την αλγεβρική δομή τους ή, αντίθετα, να αναπαρασταθούν από ένα σχήμα πλούσιας αλγεβρικής δομής.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αλγεβρική Γεωμετρία

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αλγεβρικά σύνολα, Αλγεβρικοποίηση, CAGD-CAE, Απαλοίφουσες, Συζυγίες

ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Μας ενδιαφέρει η αναπαράσταση γεωμετρικών αντικειμένων, όπως επιφάνειες και καμπύλες. Η ποικιλομορφία αυτού που μπορεί να ονομαστεί γεωμετρικό αντικείμενο εξαρτάται από το είδος των μετασχηματισμών που θα πρέπει να είναι δυνατές σε αυτά τα αντικείμενα και από το βαθμό πολυπλοκότητας που τις χαρακτηρίζει. Η χρήση άλγεβρας αποδείχθηκε ένας πολύ ισχυρός τρόπος αντιμετώπισης ενός ευρέος φάσματος αντικειμένων. Έτσι τα περισσότερα γεωμετρικά αντικείμενα που μελετήθηκαν εδώ είναι αλγεβρικά αντικείμενα. Αν μερικές φορές αναφερόμαστε σε μη αλγεβρικά αντικείμενα, είναι πάντοτε με την προοπτική να κατασκευαστεί μια αλγεβρική δομή από αυτά.

Για να περιγράψει ένα γεωμετρικό αντικείμενο, μπορεί κανείς να χρησιμοποιήσει το λεξιλόγιο του σχολείου, συνδυάζοντας ευθείες γραμμές, κύκλους και άλλα απλά σχήματα. Κάποιος μπορεί επίσης να καταρτίσει μια λίστα με όλα τα σημεία που ανήκουν στο αντικείμενο, μέχρι μια δεδομένη ακρίβεια. Είναι επίσης δυνατόν να περιγράψουμε ένα αντικείμενο με τη χρήση ιδιοτήτων διάκρισης που ικανοποιούνται από τα σημεία του (π.χ. είναι αμετάβλητες από ένα δεδομένο μετασχηματισμό) ή τις συντεταγμένες του (π.χ. ικανοποιώντας κάποια εξίσωση). Ή, πιο αφηρημένα, μπορεί κανείς να δώσει ιδιότητες του αντικειμένου, όπως το μέγεθός του, ο όγκος του, τα συνδεδεμένα υποσύνολα του, η περιγραφή των ορίων του, το γένος, η καμπυλότητα κλπ.

Όταν πρόκειται για υπολογιστές, όλες αυτές οι γεωμετρικές αναπαραστάσεις μπορεί να είναι χρήσιμες σε κάποιο πλαίσιο. Η κύρια διαφορά με τις αφηρημένες παραστάσεις είναι ότι η ποσότητα πληροφοριών που απαιτείται για την περιγραφή ενός αντικειμένου μέσα σε μια δεδομένη μέθοδο αναπαράστασης πρέπει να είναι πεπερασμένη. Για παράδειγμα, οι καμπύλες Bézier και τα μπαλώματα (patches) σχηματίζουν μια βάση απλών σχημάτων που μπορούν να συνδυαστούν σε πιο πολύπλοκα σχήματα στο Computer-Aided Geometric Design (CAGD). Ακόμα πιο απλά: οι τριγωνικές επιφάνειες μπορούν να συνδυαστούν σε ένα τριγωνικό πλέγμα ενός αντικειμένου 3D, το οποίο είναι χρήσιμο στη Computer-Aided Engineering (CAE). Στην απεικόνιση bitmap, τα αντικείμενα περιγράφονται εικονοστοιχείο ανά εικονοστοιχείο. Στην Αλγεβρική Γεωμετρία, ένα αλγεβρικό σύνολο είναι το σύνολο λύσεων πολυωνυμικών ή ρητών εξισώσεων. Και ούτω καθεξής…

Φυσικά, ένα αντικείμενο που αναπαρίσταται με μια μέθοδο αναπαράστασης μπορεί να μην αναπαρίσταται με άλλη μέθοδο. Μια ελλειπτική καμπύλη, για παράδειγμα, μπορεί να περιγραφεί ως το σύνολο λύσεων ενός πολυώνυμου βαθμού 3, αλλά δεν μπορεί να περιγραφεί χρησιμοποιώντας καμπύλες Bézier ή ακόμα και ρητές παραμετροποιήσεις. Από την άλλη πλευρά, ένα μεγάλο σύνολο αντικειμένων μπορεί να αναπαριστάται χρησιμοποιώντας αρκετές διαφορετικές μεθόδους αναπαράστασης. Ένα σημαντικό πρόβλημα είναι τότε να μπορέσουμε να περάσουμε από μια αναπαράσταση σε μια άλλη. Η δυσκολία μετατροπής ενός αντικειμένου από μία αναπαράσταση σε άλλη σχετίζεται με την ποικιλομορφία αυτών των μεθόδων αναπαράστασης.

Μεταξύ αυτών των μεθόδων αναπαράστασης, δύο είδη έχουν μεγαλύτερη σημασία για

εμάς:
• Παραμετρικές παραστάσεις, λαμβάνοντας μία ή περισσότερες παραμέτρους ως είσοδο και ένα σημείο του αντικειμένου ως έξοδο. Τυπικά, μια παραμετρική αναπαράσταση ενός αλγεβρικού αντικειμένου είναι μια ρητή απεικόνιση, με ρητή αντίστροφη απεικόνιση, μεταξύ ενός προβολικού χώρου και αυτού του αντικειμένου.
• Αλγεβρικές παραστάσεις, λαμβάνοντας ένα σημείο του περιβάλλοντος χώρου ως είσοδο και απαντά αν το σημείο αυτό ανήκει στο αντικείμενο ή όχι. Τυπικά, μια αλγεβρική αναπαράσταση είναι ένα σύνολο πολυώνυμων που μηδενίζονται ταυτόχρονα στο αντικείμενο.

Το έργο που παρουσιάζεται εδώ περιγράφει ενδιαφέροντες τρόπους για την αναπαράσταση αλγεβρικών αντικειμένωνκαι εστιάζει σε αλγεβρικές αναπαραστάσεις και αλγορίθμους αλλαγής αναπαράστασης με έμφαση στην αλγεβρικοποίηση, δηλαδή αλγορίθμους που εξάγουν μια αλγεβρική αναπαράσταση.

Ο συγγραφέας παρακολούθησε διδακτορικές σπουδές στο Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών στην Ελλάδα (ΕΚΠΑ) στο πλαίσιο του έργου ARCADES που χρηματοδοτήθηκε από την ΕΕ (Marie Skłodowska-Curie Actions). Στο πλαίσιο αυτών των μελετών, ο συγγραφέας πέρασε 3 μήνες στο ερευνητικό κέντρο SINTEF στο Olso της Νορβηγίας και 4 μήνες στο ερευνητικό κέντρο RISC Software GmbH στο Hagenberg της Αυστρίας. Και στις δύο περιπτώσεις, η εργασία πραγματοποιήθηκε σε συνεργασία με τοπικές ερευνητικές ομάδες.

Στο κεφάλαιο 3, αναπτύσσεται ένας νέος αλγόριθμος αλγεβρικοποίησης για ένα ειδικό είδος 3D αντικειμένων. Οι όγκοι ”σάρωσης” (swept volumes) είναι αντικείμενα κατασκευασμένα με τοπική περιγραφή της γεωμετρίας τους εφαρμόζοντας έναν ή περισσότερους μετασχηματισμούς ισομετρίας. Χτίζουμε μια δομή για την αλγεβροποίηση χρησιμοποιώντας και τα δύο αυτά συστατικά. Παρουσιάζονται επίσης δύο αλγόριθμοι αλγεβρικοποίησης που χρησιμοποιούνται στη βιομηχανία. Το κεφάλαιο αυτό βασίζεται στο άρθρο [45].

Στο κεφάλαιο 4, αναπτύσσεται ένας νέος αλγόριθμος αλγεβρικοποίσης για αλγεβρικά σύνολα συν-διάστασης αυστηρά μεγαλύτερης από 1. Ενώ αρκετοί άλλοι αλγεβρικοί αλγόριθμοι τείνουν να είναι ταχύτεροι, ο αλγόριθμος που προτείνουμε είναι καταλληλότερος για αλγεβρικά σύνολα μεγάλης συν-διάστασης. Η ιδέα πίσω από αυτόν τον αλγόριθμο προέρχεται από τη θεωρία των Chow forms. Το κεφάλαιο αυτό βασίζεται στα άρθρα [28] και [29].

Στο κεφάλαιο 5, αναπτύσσεται ένας νέος αλγόριθμος αλγεβρικοποίησης με βάση τους πίνακες. Αυτός ο αλγόριθμος βασίζεται σε συζυγίες (syzygies) και τετραγωνικές συζυγίες. Παρέχει μια πολύ ισχυρή σχέση μεταξύ των παραμετρικών και των αλγεβρικών αναπαραστάσεων, επιτρέποντας την αντιστροφή της απεικόνισης μεταξύ των παραμετρικών και των περιβαλλοντικών χώρων μέσω του πίνακα. Το κεφάλαιο βασίζεται στο άρθρο [10].

Στο κεφάλαιο 6, συγκρίνουμε διαφορετικές μεθόδους αλγεβρικοποίησης, υπογραμμίζοντας τα πλεονεκτήματα και τα μειονεκτήματα της καθεμιάς. Επειδή οι περισσότερες αναπαραστάσεις μπορούν να είναι πιο χρήσιμες από τις άλλες ανάλογα με την εφαρμογή (γεγονός που ισχύει όταν συγκρίνουμε τις παραμετρικές με τις αλγεβρικές αναπαραστάσεις, αλλά ακόμη και όταν συγκρίνουμε τις αλγεβρικές αναπαραστάσεις μεταξύ τους), περισσότερη δουλειά είναι απαραίτητη ώστε να επιλεχθούν οι βέλτιστες.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Ioannis Emiris for doing me the honour of working under his direction and the joy of being given the possibility to work in Athens. I may not emphasise enough the chance I was given to spend more than three years in such good conditions for doing a PhD thesis with him. I would also like to thank Christos Konaxis, who has been of a great help in many ways.

I would like to thank my colleagues at the Erga lab of the University of Athens: Ioannis Psarros, Anna Karasoulou, Emmanouil Christoforou, Apostolos Chalkis, Christina Katsamaki, Evangelos Anagnostopoulos, Apostolos Florakis and, last but not least, Evangelos Bartzos, with which I was very happy to spend time and discuss of various things. The diversity of topics in mathematics and informatics done at Erga was a great experience, as well as the possibility to share our works.

I would like to thank my fellows from the ARCADES group. Thanks again to Vangelis - who was with me in Athens -, Theofanis Katsoulis, Sotirios Chouliaras, Konstantinos Gavriil, Jan Legerský - best wishes for Jana and him -, Yairon Cid Ruiz, Fatmanur Yıldırım - and the time we have spent at INRIA and with Jan at various (mostly sloping) places -, Alvaro Fuentes-Suarez, Ahmed Blidia, Francesco Patrizi, Andrea Raffo - and the time we have spent in Oslo and working together in Hagenberg - and Michael Jimenez. Although we are geographically far from each other, we have tied a special bound by reuniting regularly.

I would like to thank Oliver Barrowclough, Georg Muntingh and Alexander Leutgeb for their hospitality and their help during the time I have spent in Oslo (Oliver and Georg) and Hagenberg (Alexander). Many thanks also to the team of the INRIA at Sophia-Antipolis: Evelyne Hubert, Bernard Mourrain for their help and Laurent Busé for his help and essential collaboration with Fatmanur and me.

Thanks to my family for their support.

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 675789. It has been done at the University of Athens in Greece in collaboration with Athena Research and Innovation Centre.

The ARCADES and Erga websites can be found at the following addresses:
`http://arcades-network.eu/`
`http://erga.di.uoa.gr/`

Marie Skłodowska-Curie Actions

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

We are interested in the representation of geometric objects, such as surfaces and curves. The diversity of what can be called a geometric object depends on what kind of operations should be possible on these objects and the degree of complexity we consent to them. Using algebra proved to be a very powerful way of dealing with a broad range of objects; thus most of the geometric objects studied here are algebraic objects. If we sometimes refer to non-algebraic objects, it is always with the prospective of constructing an algebraic structure out of them.

In order to describe a geometric object, one can use the vocabulary of the school, combining straight lines, circles and other simple shapes. One can also draw up a list of all the points belonging to the object, up to a given precision. It is also possible to describe an object by the use of discriminating properties satisfied by its points (e.g. being invariant by a given transformation) or its coordinates (e.g. satisfying some equation). Or, more abstractly, one can give properties of the object, such as its size, compactness, connected components, description of its boundaries, genus, curvature, etc.

When it comes to computers, all these geometric representations can be useful in some context. The main difference with abstract representations is that the quantity of information required to describe an object within a given representation method must be finite. For example, Bézier curves and patches form a basis of simple shapes that can be combined to have more complex shapes in Computer-Aided Geometric Design (CAGD). Even simpler: polygonal surfaces can be combined in order to for a polygonal mesh of a 3D object, which is useful in Computer-Aided Engineering (CAE). Also useful for CAE, Algebraic Geometry defines a shape as the zero set of polynomial or rational equation(s). In bitmap imaging, objects are described pixel by pixel. And so on…

A representation is thus a list of informations (typically numbers) while a representation method determines how these informations should be interpreted. More theoretically, a (computer) representation method is a partial surjection from $\mathbb{N}^{\mathbb{N}}$ to a set of representable objects by that representation method. This set of representable objects can have at most the cardinality of the continuum, which is large enough for representing most of geometric objects one can think of. For instance, the sets of (graphs of) polynomials, analytic functions over the unit complex disc or even continuous functions over a compact subset of $\mathbb{R}^n$ are all representable sets (see [54, 55] for more details about these representation methods of very large spaces). However, the set of (graphs of) real-valued functions is not representable. In our algebraic framework, we restrain ourselves mostly to representation methods of algebraic varieties, the geometric objects we can build out of them and the operations that can be done with them.

Of course, an object representable by one representation method may not be representable by another method. An elliptic curve, for instance, can be described as the zero set of a degree 3 polynomial but it cannot be described using Bézier curves or even rational parameterizations. On the other hand, a large set of objects can be represented using several different representation methods. An important problem is then to be able

to pass from a representation to another. The difficulty to convert an object from one representation to another relates with the diversity of these representation methods.

Amongst these representation methods, two kinds are of greater importance for us:
• Parametric representations, taking one or several parameters as input and a point of the object as output. Typically, a parametric representation of an algebraic object is a birational map between a projective space and that object.
• Implicit representations, taking a point of the ambient space as input and outputting whether that point belongs to the object or not. Typically, an implicit representation is a set of polynomials that simultaneously vanish on the object.

In this thesis, we sometimes refer to the *generic case*. Unless explicitly stated otherwise, a property is said to hold for the generic case when it holds for a (topologically) dense subset of objects and also holds (probabilistically) almost surely when a random object is picked with a dense probability measure. In algebraic geometry, these two notions of density and almost certainty coincide most of the time, provided that the underlying field we work in is algebraically closed (typically, $\mathbb{C}$).

The work presented here is about (1) describing interesting ways to represent algebraic objects with a focus on implicit representations and (2) developing algorithms to switch from a representation to another with a focus on implicitization, i.e. algorithms that output an implicit representation.

The defender's web-page can be found at `http://users.uoa.gr/~claroche/` on the website of the University of Athens.

1.1 Parametric and implicit representations

Our study is restricted to rings and fields of characteristic 0, typically $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$. Algebraic objects are subsets of a module, a vector space, an affine space or a projective space of dimension n, called the ambient space, and usually denoted by $\mathbb{K}^n$, Aff$(\mathbb{K}^n)$ or $\mathbb{P}^n(\mathbb{K})$ depending on the circumstances. The algebraic closure of $\mathbb{K}$ is $\overline{\mathbb{K}}$.

We use the variables x_i for the coordinates of points in the ambient space (or x, y or x, y, z if its dimension is 2 or 3). We us the variables t_j for the coordinates of points in the parametric space (or t or s, t if its dimension is 1 or 2).

1.1.1 Varieties

Definition 1. *An* algebraic variety V *in* $\mathbb{K}^n$ *is the set of common solutions of polynomials* $F_1, \ldots, F_c$ *in* n *variables* $x_1, \ldots, x_n$ *where* $c, n \in \mathbb{N}^*$*:*

$$V := \mathcal{Z}(F_1, \ldots, F_c) = \{x_1, \ldots, x_n \mid F_i(x_1, \ldots, x_n) = 0, 1 \leq i \leq c\} \tag{1.1}$$

The space $\mathbb{K}^n$ *is called the ambient space and the set of polynomials* $\{F_1, \ldots, F_c\}$ *is an implicit representation of* V*.*

A projective variety V' *in* $\mathbb{P}^n(\mathbb{K})$ *is defined in a similar way:*

$$V' := \{(x_0 : \cdots : x_n) \in \mathbb{P}^n(\mathbb{K}) \mid G_i(x_1, \ldots, x_n) = 0, 0 \leq i \leq c\}$$

where $G_0, \ldots, G_c$ *are homogeneous polynomials in* $n+1$ *variables.*

Varieties are very interesting because they provide an convenient theoretical framework when representing geometric objects. They concern representations extensively used in CAGD, computer graphics and simulations, such as Bézier curves or surfaces, splines and all their variations, etc. The algebraic operations having geometric interpretations, they are easier to manipulate than analytic representations. The algebraic structure of varieties also comes together with object invariants: properties such as dimension or genus that can be used to classify the different kinds of varieties. Here are some of these classifying properties:

Definition 2. *A variety is said to be* irreducible *if it is generated by a prime ideal* $< F_1, \ldots, F_c >$*. Geometrically, it means that it can not be split in two non-empty varieties.*

The dimension d *of a variety* V *is the maximal length minus one of a strictly increasing sequence of non-empty irreducible subvarieties of* V*,* $V_0 \subsetneq V_1 \subsetneq \cdots \subsetneq V_d \subset V$*. It is also the maximal dimension of its tangential vector spaces. If* $d = n-1$ *and all of its maximal irreducible subvarieties have the same dimension* d*, then* V *is said to be a* hypersurface*.*

The codimension *of a variety is the dimension of the ambient space minus its own dimension,* $n-d$*. A variety needs at least* $n-d$ *equations for any of its implicit representation.*

The degree $\deg(V)$ *of* V *is the number of intersection of* V *with a generic linear subspace of dimension* $n-d$*, over an algebraically closed extension of* $\mathbb{K}$ *(typically* $\mathbb{C}$*). When* V *is a hypersurface, it is generated by a radical principal ideal* $< F >$ *and its degree is the total degree of* F*.*

The genus g *of a smooth complex variety* V *of dimension* d *is the number of linearly independent holomorphic* d*-forms on* V*:*

$$g = \dim(H^0(V, \Omega^d))$$

where Ω^d *is the space of holomorphic* d*-forms and* H^0 *is the relative singular cohomology. The definition extends to non-smooth complex varieties by decreeing that the genus is invariant under birational maps (it is already invariant under birational maps between smooth varieties).*

The Riemann-Roch theorem implies that the definition of the genus g is equivalent to $g := \frac{(\deg(V)-1)(\deg(V)-2)}{2} - s$ in the case of complex curves, where s is the sum of the multiplicities of the singular points (see [35, Chapter 4]).

While the genus is an important algebraic invariant, the degree of varieties is much more important for us. Its usage is specified by another fundamental result: the theorem of Bézout.

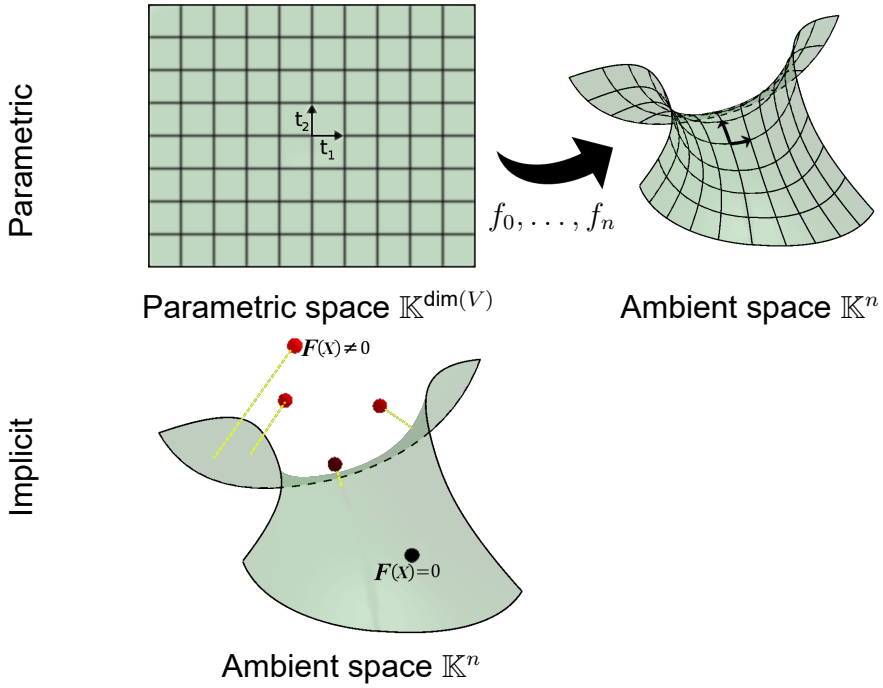


Figure 1.1: Parametric representation versus implicit representation

Theorem 1. (Bézout's theorem) [15, Chapter 8, §7] *Let* V_1 *and* V_2 *be two projective varieties in an algebraically closed field without common components, i.e. such that for all the maximal irreducible components* $W_1 \subset V_1, W_2 \subset V_2$*, we have* $\text{codim}(W_1 \cap W_2) = \text{codim}(W_1) + \text{codim}(W_2)$ *if* $\text{codim}(W_1) + \text{codim}(W_2) \leq n$ *and* $W_1 \cap W_2 = \emptyset$ *if* $\text{codim}(W_1) + \text{codim}(W_2) > n$*.*
Then $\deg(V_1 \cap V_2) = \deg(V_1)\deg(V_2)$*.*

This theorem can be seen as a generalisation of the fundamental theorem of algebra that links the number of roots of a polynomial to the degree of that polynomial.

Definition 3. *A* rational parametric representation *or a* rational parameterization *of a variety* V *(resp. projective variety* V'*) is a list of rational polynomials* $\frac{f_1}{f_0}, \ldots, \frac{f_n}{f_0}$ *in* $\dim(V)$ *variables (resp. homogeneous polynomials* $q_0, \ldots, q_n$ *of the same degree in* $\dim(V') + 1$ *variables) such that:*

$$V = \overline{\left\{ \left(\frac{f_1(t_1, \ldots, t_d)}{f_0(t_1, \ldots, t_d)}, \ldots, \frac{f_n(t_1, \ldots, t_d)}{f_0(t_1, \ldots, t_d)} \right) \mid (t_1, \ldots, t_d) \in \mathbb{K}^d, f_0(t_1, \ldots, t_d) \neq 0 \right\}}$$
$$\textit{resp. } V' = \left\{ (q_0(t_0 : \cdots : t_d) : q_1(t_0 : \cdots : t_d) : \cdots : q_n(t_0 : \cdots : t_d)) \mid (t_0 : \cdots : t_d) \in \mathbb{P}^d(\mathbb{K}) \right\} \quad (1.2)$$

A proper parameterization *of* V *is a rational parameterization for which almost all the points of* V *has only one preimage.*

A rational variety *is a variety having at least one rational parametric representation.*

1.1.2 Switching of representation

Before describing why using different kinds of representation may be useful, let us notice that rational varieties form a strict subset of varieties. In dimension 2 already, we have the following characterization.

Proposition 2. *In $\mathbb{C}^2$ or $\mathbb{P}^2(\mathbb{C})$, a curve is rational if and only if it is irreducible and its genus is 0.*

This follows directly from the invariance of genus through birational maps and the fact that the genus of $\mathbb{P}^1(\mathbb{C})$ is 0. Using the Riemann-Roch theorem, one can explicitly construct a rational parameterization of a singular complex curve by blowing-up repeatedly its singular points. When the genus is 0, i.e. the singularities are many enough (counted with multiplicity), a resolution of singularities leads to a birational map between the curve and a line or conic easily parameterizable by $\mathbb{P}^1(\mathbb{C})$ (see [62] for details).

This idea of using resolutions of singularities of varieties in order to parameterize them can be generalised to varieties other than planar curves thanks to a theorem of Hironaka (see [34] or [32]) stating that any variety in characteristic 0 admits a resolution of singularities. This result was sought for several decades before Hironaka gave a proof, using chains of blow-ups. From that point of view, a rational parameterization is given by the chain of blow-ups, provided that the smooth variety at the end of the chain is parameterizable, that is to say the singularities are complicated enough for the variety to get simple once they are blown-up.

While hypersurfaces can be implicitly represented by a single equation, varieties of codimension $n - d \geq 2$ require at least $n - d$ implicit equations. When $n - d$ equations are enough, the variety is said to be a *complete intersection* but it is unfortunately not always the case. Thereby, the number of equations obtained with an implicitization algorithm is not always predictable beforehand. The Bézout's theorem1 states that the variety degree of a complete intersection is the product the total degrees of its generating polynomials $F_1, \ldots, F_c$ (taking the radical ideal of $< F_1, \ldots, F_c >$ if necessary). A classical example of a variety that is not a complete intersection is the twisted cubic: $\{(s^3 : s^2\,t : s\,t^2 : t^3)\}_{(s:t)\in\mathbb{P}^1(\mathbb{C})}$. Indeed, the twisted cubic is a non-planar space curve of degree 3. However, considering two surfaces S_1, S_2 in $\mathbb{P}^3(\mathbb{C})$ that are not planes and have no common components, their intersection $S_1 \cap S_2$ is a curve a degree $\deg(S_1)\deg(S_2) \geq 4$, by the Bézout's theorem. Thus the twisted cubic is not the intersection of two surfaces. It is actually the intersection of three surfaces in $\mathbb{P}^3(\mathbb{C})$ (see figure 1.2).

Example 1. *$\mathcal{Z}(\{xy\}) = \{(x, y) \in \mathbb{R}^2 \mid x = 0$ or $y = 0\}$ is not irreducible and thus not a rational variety.*

$\mathcal{Z}(\{y^2 - x^3 + x\})$ is a smooth elliptic curve thus its genus is non-zero and thus it is not a rational variety, both in $\mathbb{C}^2$ and $\mathbb{R}^2$.

We end this section with considerations about the relations between the degrees of parameterizations, implicit polynomials and the degree of the variety.

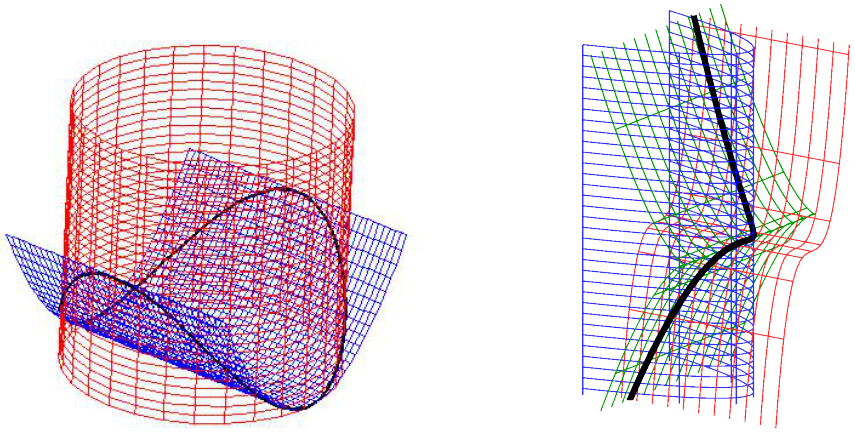


Figure 1.2: Two curves: a complete intersection of degree 4 (*left*) and the twisted cubic which is not a complete intersection (*right*).

Given a proper parameterization $\frac{f_1}{f_0}, \ldots, \frac{f_n}{f_0}$ of a variety V, the degree of V is bounded by $\deg(V) \leq \prod_{i=1}^{d} \max_j(\deg_{t_i}(f_j))$. For curves, the inequality becomes an equality and $\deg(V) = \max_j(\deg f_j)$. Also, in the homogeneous setting of a proper parameterization $q_0, \ldots, q_n$ of V', the situation is more symmetric and $\deg(V') = \deg(q_0)^d$. Hypersurfaces always have an implicit polynomial F such that $\deg(F) = \deg(V)$. However, for varieties of codimension 2 or more, it is sometimes possible to obtain an implicit representation made of polynomials that all have a degree strictly lower than $\deg(V)$. Chapters 2 and 5 give different proofs of this fact. In Chapter 5, it is shown that for a generic space curve, we have $\max_i(\deg(F_i)) = \lceil \frac{2\deg(V)}{3} \rceil$ for some implicit representation $\{F_i\}_i$.

1.2 Interest in applications

In this section, the advantages and disadvantages of each kind of representation is discussed. Also, while 2D and 3D objects are the core objects of the study, the algebraic tools developed can be applied to other situations, in particular when higher dimensional spaces are involved. For instance, consider to be given a huge amount of data about weather conditions over an area (position, wind speed and direction, sunlight, humidity, time of day, etc.). The existence of relations between these data, if not algebraic, can be approximated by algebraic relations through an implicitization (or e.g. a statistical regression analysis) and thus describe an algebraic variety. A parameterization of such variety would then allow to span the range of plausible weather data, enabling the study of extremal conditions or the reconstruction of a plausible variation between two given states. In such a situation, the ambient space dimension n is the number of weather conditions measured and can be quite high.

The big picture is that parametric representations are useful when sampling and object while implicit representations are useful when looking for the relative position of a given point with respect to the object.

Using **parametric representations** has the following advantages and disadvantages:

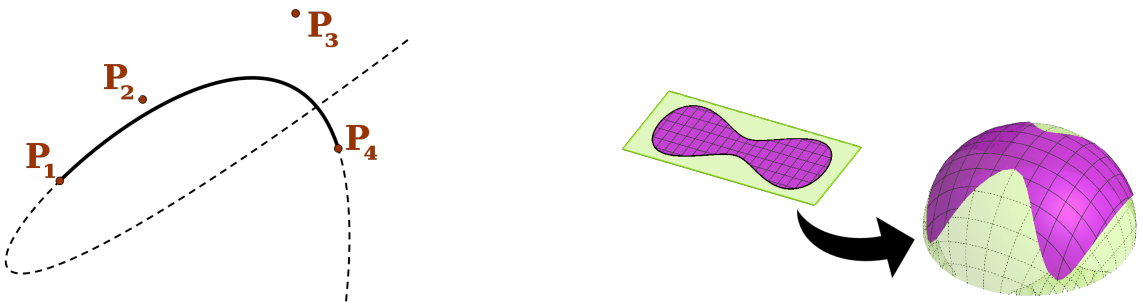


Figure 1.3: Problems with global representations: trimming a Bézier curve to its control polygon may keep unwanted parts of the global curve (*left*), trimming on the parametric space cannot be carried to a global representation, unless an inverse map from the ambient space to the parametric space is available (*right*)

- Sampling points on the object is very easy.
- Displaying the object on a screen is fast.
- Trimming the object to only consider a local portion of it is possible.
- More precisely, trimming can be done both on the parametric space or in the ambient space.
- They are preferred representations for additive manufacturing.
- The intersections of rationally parameterized objects (e.g. 3D surfaces) are not always rationally parameterized.

Using **implicit representations** has the following advantages and disadvantages:

- Checking the membership of a query point is very easy.
- The intersection of several implicitly represented objects is simple to handle.
- The intersection of an implicitly represented object with a parametrically represented object is usually simple.
- They provide geometric and algebraic informations (genus, ideal…).
- They allow ray-tracing methods, which enables high-quality rendering of the objects.
- They are preferred representations for subtractive manufacturing.
- They describe the variety globally; trimming can be done in the ambient space but not the parametric space (see Fig. 1.3).

From these lists of features, it appears that parametric and implicit representations are complementary. Except for specific applications, having both representations of a single object is usually the best way to go.

In practice, the whole varieties are not interesting when dealing with geometric objects: only more or less small parts of each variety are considered and gathered to form a piecewise-algebraic object. Parametric representations allow to cut the parts of varieties in the space of parameters, considering for instance only the unbroken segment of the Bézier curve shown in figure 1.3 or the purple part of the patch in that same figure. Both representations allow to cut varieties in the ambient space, considering for instance only the part of a curve inside a rectangular box or a polygon. The problem presented in figure 1.3 is that it is not always easy, let alone possible, to discriminate in the ambient space a part of a variety trimmed in a parametric space.

Both representations allow rendering algorithms. Ray-tracing methods for rendering implicitly-represented objects can handle reflections and lightning accurately, which makes them suitable for very high-quality rendering. However, the speed when displaying parametrically-represented objects is outmatched and can make the difference between a real-time rendering and a non-real-time rendering. Because of the speed advantage, designers use parametric representations to produce free-form objects very easily in practice. Then, if needed, other representations must be computed. For instance, CAE engineers need another representation (implicit or mesh) in order to compute the objects' hardness, flexibility or, more generally, its physical properties and behaviour (which consists in solving partial differential equations most of the time).

On top of that, a lot of 3D objects are represented using polygonal meshes (triangular meshes for most of it). These can be considered both as parametric representations or implicit representations since it only consists of linear surfaces, from which both representations are immediate to construct. Indeed, given three points A, B and C, the triangle ABC is:

i. parametrically represented by $f(s,t) = A + s\overrightarrow{AB} + t\overrightarrow{AC}$, with $0 \leq s,t$ and $s+t \leq 1$,

ii. implicitly represented by $F(x,y,z) =< (x,y,z) \cdot (\overrightarrow{AB} \times \overrightarrow{AC}) > - < A \overrightarrow{(} \overrightarrow{AB} \times \overrightarrow{AC}) >$, with (x,y,z) lying in the three half-spaces defined by the triangle's side and its normal.

Meshes are more than enough to represent very simple objects (i.e. with flat surfaces). They are not efficient any more when it comes to represent curved shapes, as many polygons are required for them for a result that is not as smooth as what we could ask for. We need to use more accurate representations when meshes are not a satisfying solution, and thus have algorithms for manipulating them and switch from a representation to another with the minimal loss of precision.

The algorithm of marching cubes (see [47]) can be used to compute a mesh out of an implicit representation. Using it, an approximation of implicitly-represented objects can be displayed relatively fast, though slower than when a parametric representation is directly available.

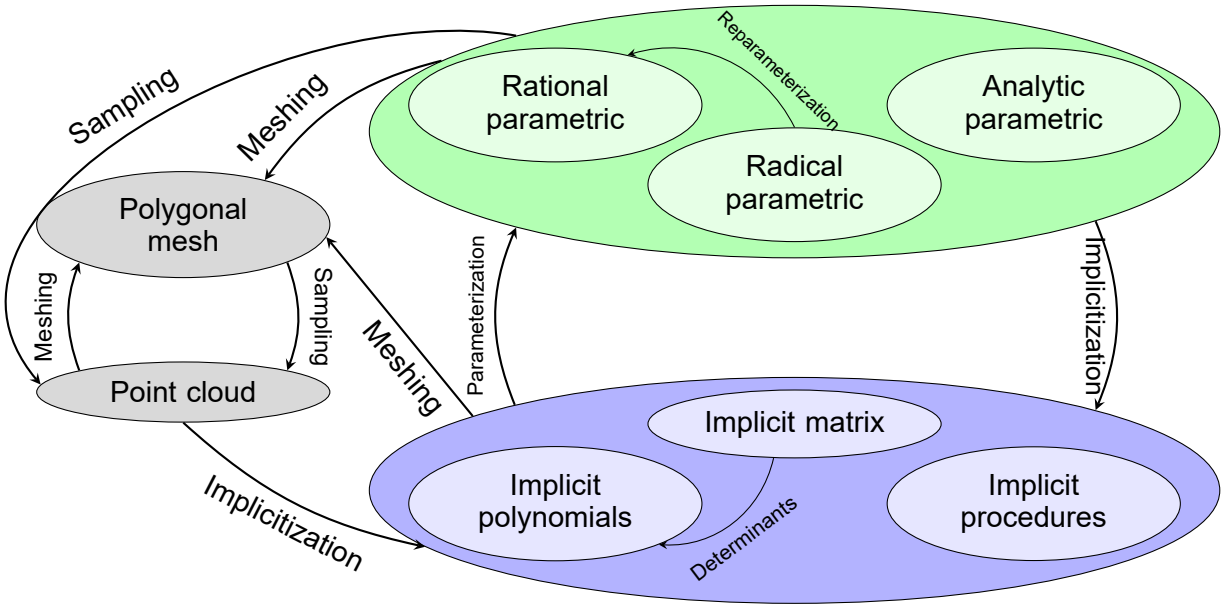


Figure 1.4: Different representations of varieties

Another way to represent 3D objects, closely related to meshes, is the combinations of basic shapes. Spheres, cylinders, ellipsoids or cones, for instance, are not linear surfaces like triangles but they are still easy to convert to parametric or implicit forms. They can give a nice middle ground between curved shapes and simplicity. There are methods for producing basic shapes from point clouds with exact fitting (e.g. [9]) or with approximate fitting (e.g. [58]).

One of the biggest issues with parametric representations is the fact that the intersection of two rationally parameterized objects (typically two surfaces) are not necessarily rational varieties (the intersection curve(s) cannot be rationally parameterized). This is a huge problem to deal with in CAE and in particular when switching back and forth from parametric representations produced by CAGD to implicit representations or meshes required in CAE.

For instance, consider a simple object made of two pieces. Each piece being produced by a CAGD designer, they are parametrically represented. When assembled together, one needs to cut them along their intersection. Since this cannot always be done, only an approximation of the intersection is computed, which may lead to small gaps in-between or, on the contrary, small parts of the pieces getting inside each other. Then, when implicitizing or meshing the object, these gaps must be fixed since the object would not be watertight otherwise and thus show misleading physical properties (see the figure 1.5).

The problem shows up again in the other direction if, once the CAE engineers have adjusted the object's shape to optimize its physical properties, a CAGD-compatible representation must be made out of the new shape for further design purpose (e.g. place that simple object as a part of a bigger one).

Matrix representations are a fitting way to represent a wide range of varieties. Instead of

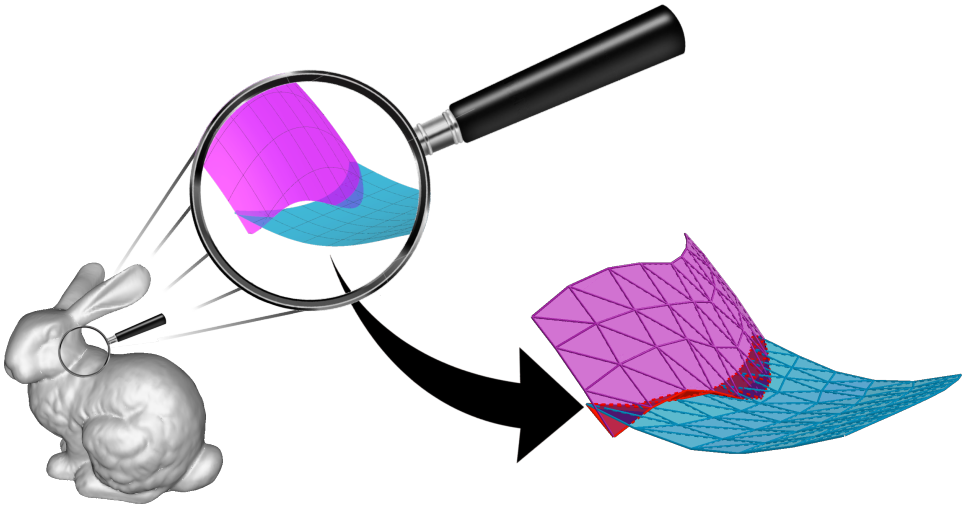


Figure 1.5: The intersection of parametric surfaces may not be represented accuratly in a parametric form, generating gaps or surfaces crossing each others. The mesh can be fixed by collapsing the transition between the surfaces (in red) at the cost of data loss and thus problems when converting the mesh back to parametric patches.

considering a variety as the set of common zeros of implicit polynomials, it is described as the set of points dropping the rank of a formal matrix. For hypersurfaces, we use a square matrix and the drop of rank property is equivalent to the vanishing of the determinant (which is then the variety's unique defining polynomial). For varieties of lower dimensions, we use a rectangular matrix and the drop of rank property is equivalent to the vanishing of all of that matrix's largest minors. We develop matrix representation algorithms in the section 2.3 and Chapter 5. The matrix representation developed in Chapter 5 also provide a nice solution to the problems shown in figure 1.3.

A point cloud is a set of scattered points of a hypersurface (curve in 2D, surface in 3D, etc), possibly given with their normals or other data. They are impractical and give few informations about the variety. However, raw data are often obtained in that form, for instance with 3D cameras using lasers to estimate the depth of an object in a given view frame. Thus, many algorithms exist in order to structure point clouds:

- produce a mesh for them, which is a combinatorial task that is not as simple as it may look (see e.g. [1, 74]),
- try to fit them to basic shapes like cylinders or cones (see e.g. [9, 58]),
- produce implicit surfaces, either exact or approximate (see Chapter 3),
- interpolate a parametric hypersurface, either exactly or approximately (see section 2.3).

Voxels have been used as an alternative to point clouds. They allow efficient volumetric computations but are usually more expensive storage-wise due to holding data indexed on space coordinates. Voxels, much like bitmap images, are out of the range of our work.

Finally, radical and analytic parameterizations are parameterizations allowing more operations than the 4 basis operations used by rational parameterizations. Radical parameterizations allow square root operations on polynomials and, more generally, n-th root operations. Analytic parameterizations allow the use of functions like $\exp, \cos, \sin$ and, more generally, functions that can be defined with integrals or limits. The latter ones having few to none algebraic properties, they are extremely impractical regarding the algebraic tools developed in this work. While some analytic parameterizations are describing rational varieties, the differences of the parameterizations are striking even in these very special case (e.g. the circle, which can be parameterized both as $(\cos(t), \sin(t))$, which is a bijective geodesic map from $[-\pi, \pi)$ to the circle, and $\left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}\right)$, which is a birational map with $\mathbb{P}^1$). Algebraic tools are also hard to apply to radical parameterizations but they may be reparameterized with rational parameterizations when they describe branches of rational varieties (see [61]). The elliptic curve of example 1 admits a radical parameterization but cannot be reparameterized that way.

The evolution of mathematical methods for parameterization or implicitization is sometimes little known by the communities of computer engineers using such methods. It was brought to our attention that textbooks in computer graphics [33] sometimes share a belief according to which there is no implicitization algorithm that would work for all situations. While it is true that implicitization algorithms should be chosen for their efficiency in a specific situation (there is no implicitization algorithm that consistently outbests all the others), many algorithms require little to no hypothesis on the variety that should be implicitized. On top of that, these hypotheses can sometimes be worked around: for instance, Gröbner bases algorithms may require the parameterization to admit no base point. But even then, one can bypass this requirement using a trick, namely saturation of the equations for this example. For some other algorithms, a small perturbation of the variety that keeps its algebraic invariants unmodified can be enough for the algorithm to apply and output implicit equations of the perturbed variety. It is then possible to retrieve implicit equations of the original variety by using homotopy, i.e. finding the perturbation of the implicit equations that fits the perturbation that we applied, etc. The room for improvement in implicitization is thus not about relaxing the hypotheses but on the contrary to design algorithms that are more fitted to specific situations.

1.3 Summary and contributions of the thesis

This thesis presents results and algorithms in the field of implicit representations and implicitization algorithms.

The author has followed Ph.D. studies at the National and Kapodistrian University of Athens in Greece in the framework of the project ARCADES funded by the Marie Skłodowska-Curie Actions. As part of this studies, the author has spent 3 months at the research centre SINTEF in Oslo, Norway, and 4 months at the research centre RISC Software GmbH in Hagenberg, Austria. In both occasions, work has been done collaboratively with the local research teams.

In Chapter 2, basic tools are presented. Gröbner bases and resultants are the main tools when it comes to implicitization algorithms. We present both, with an emphasize on resultants because the algorithms on the subsequent chapters rely on them rather than on Gröbner bases. Moreover, a C++ implementation of a sparse resultant algorithm has been developed by the author based on an existing Maple implementation. We also present a very basic interpolation matrix that can be built for varieties of any (co)dimension.

In Chapter 3, a new implicitization algorithm for a special kind of 3D objects is developed. Two implicitization algorithms used in the industry are also presented. That chapter is based on the technical report [45].

In Chapter 4, a new implicitization algorithm of varieties of codimension strictly greater than 1 is developed. While several implicitization algorithms fare better on hypersurfaces, this one on the contrary is better suited for varieties of high codimension. The idea behind that algorithm comes from the theory of Chow forms which is also presented there. That chapter is based on the article [29].

In Chapter 5, a new matrix-based implicitization algorithm is developed. This algorithm relies on syzygies and chain complexes. It provides a very strong link between parametric and implicit representations, allowing to reverse the map between the parametric and ambient spaces through the implicit matrix. That chapter is based on the articles [28] and [10].

In Chapter 6, we compare different implicitization methods, outlining the advantages and drawbacks of each one. Since most representations can more useful than the others depending on the situation (a fact that is true when comparing parametric and implicit representations but even when comparing implicit representations amongst themselves), it is a necessary work to be done.

2. PRELIMINARIES

When it comes to exact implicitization, there are traditionally two frameworks: Gröbner bases and resultants.

2.1 Gröbner bases

Gröbner bases were invented, as the name doesn't suggest, independently by Hironaka in 1964 [34] (while proving that any variety in characteristic 0 admits a resolution of singularities) and Buchberger (Gröbner's student) in 1965 [5].

Even though they play only a little role in the work presented here, since we use resultants rather than Gröbner bases, they play an important role, at least historically, in implicitization in general, hence the following brief introduction.

We recall that an ideal $\mathcal{I}$ of a ring R is a subset of that ring stable under the internal sum (i.e. $\forall a, b \in \mathcal{I},\ a+b \in \mathcal{I}$) and the external product (i.e. $\forall k \in R, \forall a \in \mathcal{I},\ ka \in \mathcal{I}$). The rings we are interested in here are polynomial rings of several variables.

Definition 4. *A* monomial order *is a total order $\leq$ on the monomials respecting the product: $a \leq b \implies ac \leq bc$ for any monomial c. Since it is a total order, each polynomial has a leading term w.r.t. this monomial order.*

A Gröbner basis *of an ideal $\mathcal{I} \subset \mathbb{K}[x_1, \ldots, x_m]$ w.r.t. a specific monomial order is a list of polynomials $(P_1, \ldots, P_k)$ satisfying:*

1. *$\mathcal{I}$ is the ideal generated by $\{P_1, \ldots, P_k\}$, that is $\mathcal{I} =< P_1, \ldots, P_k >$,*
2. *the leading term of any polynomial in $\mathcal{I}$ is divisible by the leading term of P_i for some $i \in \{1, \ldots, k\}$,*
3. *the ideal of the leading terms of polynomials in $\mathcal{I}$ equals the ideal generated by the leading terms of $P_1, \ldots, P_k$,*

Note that (1. and 2.) $\iff$ 3.

There are several algorithms for computing Gröbner bases; some of them are described in [14, 15] for instance.

Most common monomial orders are the lexicographic order (the natural order on $\mathbb{K}[x_1, \ldots, x_m] = \mathbb{K}[x_m][\ldots][x_1]$), the graded lexicographic order (monomials are ordered w.r.t. their degree and w.r.t. the lexicographic order for those of the same degree) and the elimination orders. The latter ones are of most interest for implicitization; they apply to polynomial rings of type $\mathbb{K}[t_1, \ldots, t_d, x_1, \ldots, x_n]$ where two blocks of variables are distinguished. In elimination orders, the monomials $T_a X_a$ and $T_b X_b$, with $T_a, T_b \in \mathbb{K}[t_1, \ldots, t_d], X_a, X_b \in \mathbb{K}[x_1, \ldots, x_n]$, are first compared by comparing T_a and T_b w.r.t. a monomial order on $\mathbb{K}[t_1, \ldots, t_d]$ and then

in case of $T_a = T_b$ by comparing X_a and X_b w.r.t. a monomial order on $\mathbb{K}[x_1, \ldots, x_n]$. This guarantees that monomials containing no variable amongst $t_1, \ldots, t_d$ are smaller than any monomial containing a variable amongst $t_1, \ldots, t_d$.

Such an elimination order allows to eliminate a set of variables from polynomial equations. Indeed, a Gröbner basis of an ideal $\mathcal{I} \subset \mathbb{K}[t_1, \ldots, t_d, x_1, \ldots, x_n]$ w.r.t. an elimination order is able to compute $\mathcal{I} \cap \mathbb{K}[x_1, \ldots, x_n]$ as the basis $(P_1, \ldots, P_k)$ is split into the polynomials $P_1, \ldots P_{k'}$ containing no variable $t_1, \ldots, t_d$ and the polynomials $P_{k'+1}, \ldots P_k$ all of which contain at least one variable $t_1, \ldots, t_d$.

Let V be a rational variety, $V = \overline{\left\{ \left(\frac{f_1(t_1,\ldots,t_d)}{f_0(t_1,\ldots,t_d)}, \ldots, \frac{f_n(t_1,\ldots,t_d)}{f_0(t_1,\ldots,t_d)} \right) \right\}_{(t_1,\ldots,t_d) \in \mathbb{K}^d}} \subset \mathbb{K}^n$. If V is a curve, that is $d = 1$, and the polynomials $f_0, \ldots, f_n$ are coprimes, then V is the set of points $x_1, \ldots, x_n$ verifying the following equations for some $t \in \mathbb{K}$.

$$\begin{cases} Q_1(t, x_1, \ldots, x_n) = x_1 f_0(t) - f_1(t) = 0 \\ Q_2(t, x_1, \ldots, x_n) = x_2 f_0(t) - f_2(t) = 0 \\ \ldots \\ Q_n(t, x_1, \ldots, x_n) = x_n f_0(t) - f_n(t) = 0 \end{cases}$$

A Gröbner basis of $\mathcal{I} =< Q_1, \ldots, Q_n >$ w.r.t. an elimination order thus outright gives us implicit equations of V: they are the polynomials $P_1, \ldots, P_{k'}$ generating $\mathcal{I} \cap \mathbb{K}[x_1, \ldots, x_n]$.

When V is a rational variety of dimension $d \geq 2$, it is not enough to compute the Gröbner basis of $\mathcal{I}$ defined like above because there may be ***base points***, i.e. common roots of $f_0, \ldots, f_n$, that cannot be avoided by imposing polynomials to be coprime as it can be done for curves. In this case, we must rule out any irreducible component of $\mathcal{Z}(\mathcal{I})$ that is strictly contained in $\mathcal{Z}(p_0)$ for instance. We do that by saturating $\mathcal{I}$ by p_0:

$$\begin{aligned} \mathcal{J} &:= \left\{ q \in \mathbb{K}[t_1, \ldots, t_d, x_1, \ldots, x_n] \mid \exists k \in \mathbb{N}, q p_0^k \in \mathcal{I} \right\} \\ &= < Q_1, \ldots, Q_n, 1 - \tau p_0 > \cap \mathbb{K}[t_1, \ldots, t_d, x_1, \ldots, x_n] \end{aligned}$$

where τ is a new variable.

Using that second form, we can compute $\mathcal{J}$ with a first Gröbner basis computation w.r.t. an elimination order eliminating τ. Then implicit equations of V are obtained by computing $\mathcal{J} \cap \mathbb{K}[x_1, \ldots, x_n]$ with a second Gröbner basis computation, eliminating $t_1, \ldots, t_d$ this time.

Example 2. *Consider a rational parameterization of the sphere:*

$$x = \frac{1 - s^2 - t^2}{1 + s^2 + t^2} \qquad y = \frac{2s}{1 + s^2 + t^2} \qquad z = \frac{2t}{1 + s^2 + t^2}$$

Although there does not seem to be a common root of $1 - s^2 - t^2$, $1 + s^2 + t^2$, $2s$ and $2t$, all the points at infinity $(s : t : u)$ verifying $s^2 + t^2 = 0$ (and $u = 0$) are actually base points.

Thus a Gröbner basis of $\mathcal{I} =< x(1+s^2+t^2)-(1-s^2-t^2), y(1+s^2+t^2)-2s, z(1+s^2+t^2)-2t >$ would only consist of polynomials containing s or t and no implicit equation.

Computing a Gröbner basis of $< Q_1, \dots, Q_n, 1-\tau p_0 >$ *as defined above, we obtain 5 equations:*

$$z\,s^2 + z\,t^2 + z - 2\,t, \quad y\,t - z\,s, \quad y\,s^2 + y + z\,s\,t - 2\,s, \quad x + y\,s + z\,t - 1,$$
$$2\tau + y\,s + z\,t - 2$$

By dropping the last equation, which involves τ*, we get generators of the ideal* $\mathcal{J}$*. An other Gröbner basis computation leads to 6 other equations, the implicit equation of the sphere being amongst them.*

$$s\,x + s - y, \quad s\,y + t\,z + x - 1, \quad s\,z - t\,y, \quad t\,x + t - z, \quad t\,y^2 + t\,z^2 + x\,z - z,$$
$$x^2 + y^2 + z^2 - 1$$

While Gröbner bases are extremely powerful, they can be expensive computation-wise. In particular, the worst-case complexity of Gröbner basis algorithms is usually very high although the generic case is much faster.

2.2 Resultants

Resultants have been introduced by Sylvester in 1840 [67]. Let $P, Q \in \mathbb{K}[t]$ be two polynomials of respective degree δ_1, δ_2: $P(t) = p_0 + p_1 t + \cdots + p_{\delta_1} t^{\delta_1}$ and $Q(t) = q_0 + q_1 t + \cdots + q_{\delta_2} t^{\delta_2}$. The *Sylvester resultant* of P, Q is then defined as the following quantity:

$$\mathsf{Res}_{\delta_1,\delta_2}(P,Q) = \mathsf{Det}\begin{pmatrix} \overbrace{\begin{matrix} p_0 & 0 & \cdots & 0 \\ p_1 & p_0 & \ddots & 0 \\ \vdots & p_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & p_0 \\ p_{\delta_1-1} & \vdots & \ddots & p_1 \\ p_{\delta_1} & p_{\delta_1-1} & \ddots & \vdots \\ 0 & p_{\delta_1} & \ddots & \vdots \\ \vdots & \vdots & \ddots & p_{\delta_1-1} \\ 0 & 0 & \cdots & p_{\delta_1} \end{matrix}}^{\delta_2 \text{ columns}} & \overbrace{\begin{matrix} q_0 & 0 & \cdots & 0 \\ q_1 & q_0 & \ddots & 0 \\ \vdots & q_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & q_0 \\ q_{\delta_2} & \vdots & \ddots & 0 \\ 0 & q_{\delta_2} & \ddots & q_0 \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & q_{\delta_2-1} \\ 0 & 0 & \cdots & q_{\delta_2} \end{matrix}}^{\delta_1 \text{ columns}} \end{pmatrix} \tag{2.1}$$

For simplicity, we write $\mathsf{Res}(P,Q)$ instead of $\mathsf{Res}_{\mathsf{deg}(P),\mathsf{deg}(Q)}(P,Q)$.

The matrix of the equation 2.1 is called the *Sylvester matrix* of P, Q. It can be seen as the matrix of the map $(A, B) \in \mathbb{K}[t]_{\delta_2-1} \times \mathbb{K}[t]_{\delta_1-1} \mapsto PA + QB$ w.r.t. the canonical bases of the polynomial rings (sometimes called the power bases). Thus, $\mathsf{Res}(P,Q) = 0$ if and only if there exists non-zero polynomials A, B such that $PA + QB = 0$.

Amongst the many properties of the Sylvester resultant, the Poisson formulae are both

typical and very useful:

$$\begin{aligned}\mathsf{Res}(P,Q) &= p_{\delta_1}^{\delta_2} \prod_{\alpha\in\mathsf{RootsOf}(P)} Q(\alpha) \\ &= p_{\delta_1}^{\delta_2} q_{\delta_2}^{\delta_1} \prod_{\substack{\alpha\in\mathsf{RootsOf}(P)\\ \beta\in\mathsf{RootsOf}(Q)}} (\alpha-\beta) \\ &= (-1)^{\delta_1\delta_2}\, q_{\delta_2}^{\delta_1} \prod_{\beta\in\mathsf{RootsOf}(Q)} P(\beta)\end{aligned} \tag{2.2}$$

where the roots are taken in the algebraic closure of $\mathbb{K}$.

It can be read directly from these formulae that the Sylvester resultant vanishes when the polynomials P and Q have a common root in $\overline{\mathbb{K}}$.

It is worth mentioning that resultants can be defined for polynomials with coefficients in a ring R and their resultant will still be an element of R. A sharp analysis may determine in which structure amongst R, the fractional field of R or its algebraic closure lies each of the scalars discussed in this section. It is however not the goal here and we will stick with fields for simplicity, not rings.

Resultants are extremely useful for implicitization because they are able to eliminate a variable from equations. For example, let $V = \overline{\left\{\left(\frac{f_1(t)}{f_0(t)}, \frac{f_2(t)}{f_0(t)}\right) \mid t\in\mathbb{K}, f_0(t)\neq 0\right\}}$ be a rational plane curve, with $\gcd(f_0,f_1,f_2)=1$. Then, V is the set of points (x,y) such that $\exists t, f_0(t)x - f_1(t) = 0$ and $f_0(t)y - f_2(t) = 0$. Therefore, $\mathsf{Res}_t(f_0(t)x-f_1(t), f_0(t)y-f_2(t))$ is the implicit equation of V, where Res_t is the Sylvester resultant w.r.t. the parameter t.

This gives us a first resultant-based method for implicitization of plane curves. More generally, resultants are taken w.r.t. variables of the parametric space in implicitization algorithms because those are the ones we want to eliminate from equations.

Since Sylvester, resultant technics have been generalised.

Definition 5. *A* resultant *is an map that takes $d+1$ polynomials in d variables as input, returns a single scalar as output and verifies the following properties:*

- *the output is zero if and only if the input polynomials have a common root in an algebraically closed field extension of $\mathbb{K}$ and*
- *it is polynomial with respect to the coefficients of the input polynomials.*

In other words, $\mathsf{Res} : \Delta \to \mathbb{K}$*, where* $\Delta \subset \mathbb{K}[t_1,\dots,t_d]^{d+1}$*, is a resultant if it is a polynomial map and*

$$\mathsf{RootsOf}(\mathsf{Res}) = \{(P_0,\dots,P_d)\in\Delta \mid \exists t_1,\dots,t_d \text{ such that } \forall i, P_i(t_1,\dots,t_d)=0\} \tag{2.3}$$

Similarly, a homogeneous resultant *takes $d+1$ homogeneous polynomials in $d+1$ variables as input and satisfy the same properties, where a common root is to be understood as a root in a projective space (*$\mathbb{P}^d(\overline{\mathbb{K}})$ *or* $\left(\mathbb{P}^1(\overline{\mathbb{K}})\right)^d$*).*

The space of polynomials $\mathbb{K}[t_1, \dots, t_d]$ is thus replaced by the space of homogeneous polynomials $\mathbb{K}[t_0, \dots, t_d]^{\mathsf{hom}}$ for homogeneous resultants. The homogeneous approach is more robust in the sense that it $\mathbb{K}[t_0, \dots, t_d]^{\mathsf{hom}}_\delta$, the set of homogeneous polynomials of degree δ, is a vector space. This allows Δ to be a vector subspace of $\left(\mathbb{K}[t_0, \dots, t_d]^{\mathsf{hom}}\right)^{d+1}$. For instance, $\mathbb{C}[t, s]^{\mathsf{hom}}_2 \simeq \mathbb{C}^3 \simeq \langle a, b, c\rangle$ where an isomorphism is $(a, b, c) \mapsto at^2 + bts + cs^2$. Then,

$$\begin{aligned}\mathsf{Res}(a, b, c, a', b', c') &:= \begin{vmatrix} a & 0 & a' & 0 \\ b & a & b' & a' \\ c & b & c' & b' \\ 0 & c & 0 & c' \end{vmatrix} \\ &= a^2c'^2 - abb'c' + ac(b'^2 - a'c') + a'^2c^2 - a'b'bc + a'c'(b^2 - ac)\end{aligned}$$

is a homogeneous resultant for $\Delta = \left(\mathbb{C}[t, s]^{\mathsf{hom}}_2\right)^2$.

The same formula for non-homogeneous polynomials, which is a Sylvester resultant, vanishes on couples of polynomials of degree 1 even when they do not share a common zero ($bt + c$ and $b't + c'$ may not share a common zero unlike $w(bt + cw)$ and $w(b't + c'w)$). Thus, a valid set Δ must not contain such couples and is typically not a vector space but only a dense subset of a vector space.

Although a resultant must be polynomial, it is not always the most efficient to compute the polynomial form. Since Sylvester gave a determinantal formula for his resultant, several generalisations have been searched as determinantal formulae too.

One of the key points for generalisations of Sylvester resultants is the space Δ in which lies the polynomials and a basis of which is a suitable set of parameters. For instance, with the homogeneous polynomials $P(t, s) = p_0s^3 + p_3t^3$ and $Q(t, s) = q_1ts^2 + q_2t^2$, one could be tempted to search for a polynomial with respect to p_0, p_3, q_1 and q_2. The homogeneous version of the Sylvester resultant, however, requires to introduce the "missing" parameters: $\Delta = \langle p_0, p_1, p_2, p_3, q_0, q_1, q_2\rangle \simeq \mathbb{K}[t]_3 \times \mathbb{K}[t]_2$.

Determining the minimal set of parameters to use - that is, a suitable space Δ in order to guarantee the existence of a resultant - is one of the practical problems to explore.

2.2.1 Macaulay resultants

The Macaulay resultant is the most direct generalisation of the Sylvester resultant: it detects the existence of common roots of more than two polynomials in several variables and it expresses a resultant as a determinant of a matrix, the Macaulay matrix. The Macaulay resultant is defined for a collection of $d + 1$ polynomials $P_0, \dots, P_d \in \mathbb{K}[t_1, \dots, t_d]$ in d variables and of respective degree $\delta_0 \leq \dots \leq \delta_d$. For convenience, we note $t^k := t_1^{k_1} \dots t_d^{k_d}$ the multi-index power and $|k| := k_1 + \dots + k_d$ the ℓ^1-norm of the multi-index k. We write P_i as:

$$P_i(t_1, \dots, t_d) = \sum_{\substack{k=k_1,\dots,k_d \\ 0\leq|k|\leq\delta_i}} c_{i,k}t^k.$$

Since the number of common zeros of d generic polynomials is known to be the product of their degrees, by the Bézout theorem, we expect the resultant of $d+1$ polynomials $P_0, \dots, P_d$ to satisfy the following:

$$\forall i \in \{0, \dots, d\},\ \deg_{P_i}(\mathsf{Res}) = \prod_{j \neq i} \delta_j \tag{2.4}$$

where $\deg_{P_i}$ is the degree w.r.t. the coefficients of P_i.

The Macaulay resultant is relevant when most of the coefficients $c_{i,k}$ are non-zero. For polynomials with many zero coefficients, the sparse resultant discussed in the following section is more efficient.

Consider the power $\nu := \left(\sum_{i=0}^{d} \delta_i\right) - d$ and the set of monomials $S := \left\{t^k \mid |k| \leq \nu\right\}$. Then partition S into $S_d \cup \dots \cup S_0$ recursively as follows:

$$\begin{aligned}
S_d &:= \left\{m \in S \mid \deg_{t_d}(m) \geq \delta_d\right\} \\
S_{d-1} &:= \left\{m \in (S \backslash S_d) \mid \deg_{t_{d-1}}(m) \geq \delta_{d-1}\right\} \\
&\text{etc.} \\
S_1 &:= \left\{m \in (S \backslash \cup_{i=2}^{d} S_i) \mid \deg_{t_1}(m) \geq \delta_1\right\} \\
S_0 &:= S \backslash \cup_{i=1}^{d} S_i
\end{aligned}$$

where $\deg_{t_i}$ is the degree w.r.t. the parameter t_i.

This partition satisfies the following property (see figure 2.1 and [48]), which is to be compared with the equation (2.4).

$$|B_0| = \prod_{j \neq 0} \delta_j \text{ and } |B_i| \geq \prod_{j \neq i} \delta_j \tag{2.5}$$

We may now define the Macaulay matrix M as the matrix of size $|S| \times |S|$ with columns indexed by S and rows expressing $m\,P_0$, $\forall m \in S_0$ and $\frac{m}{t_i^{\delta_i}} P_i$, $\forall m \in S_i$, for $i \in \{1, \dots, d\}$.

Theorem 3. [48] *The resultant of $P_0, \dots, P_d$ is the quotient of* $\det(M)$ *by the minor of M obtained by omitting the rows and columns corresponding to all the reduced monomials, where a monomial $m \in S$ is said to be reduced when*

$$\begin{cases} \deg(m) = \nu \text{ and } \exists! i \in \{1, \dots, d\} \text{ such that } t_i^{\delta_i} \mid m \\ \text{or } \deg(m) \leq \nu - \delta_0 \text{ and } \forall i \in \{1, \dots, d\}, t_i^{\delta_i} \nmid m \end{cases}.$$

Example 3. *Let* $\begin{cases} P_0(s,t) = u_0 + u_1 s + u_2 t \\ P_1(s,t) = -1 + 2s + s^2 + t + s\,t \\ P_2(s,t) = -1 + 3s + s^2 + 2t - t^2 \end{cases}$ *It is called the u-resultant of P_1, P_2 because it allows to solve $P_1 = P_2 = 0$ by introducing P_0. A Macaulay matrix is the*

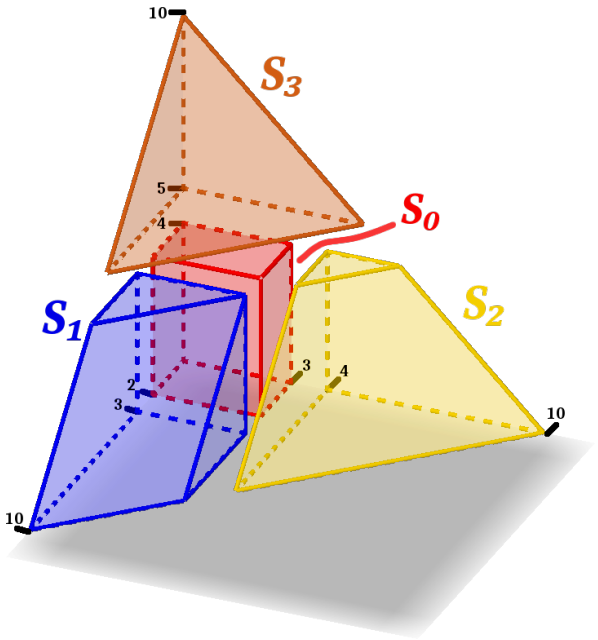


Figure 2.1: The partition of S with $d = 3$ and $\delta = (1, 3, 4, 5)$

following.

$$\begin{array}{c} \begin{array}{cccccccccc} 1 & s & t & s\,t & s^2 & s^3 & s^2\,t & t^2 & s\,t^2 & t^3 \end{array} \\ \begin{pmatrix} u_0 & u_1 & u_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_0 & 0 & u_2 & u_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_0 & u_1 & 0 & 0 & 0 & u_2 & 0 & 0 \\ 0 & 0 & 0 & u_0 & 0 & 0 & u_1 & 0 & u_2 & 0 \\ -1 & 2 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 & 1 & 1 & 1 & 0 \\ -1 & 3 & 2 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 2 & 3 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 3 & 0 & 0 & 1 & 2 & 0 & -1 \end{pmatrix} \begin{array}{c} P_0 \\ s\,P_0 \\ t\,P_0 \\ s\,t\,P_0 \\ P_1 \\ s\,P_1 \\ t\,P_1 \\ P_2 \\ s\,P_2 \\ t\,P_2 \end{array} \end{array}$$

Its determinant factorises to $(u_0+u_1-u_2)(u_0-3u_1+u_2)(u_0+u_2)(u_1-u_2)$*, which expresses the common roots of* P_1 *and* P_2*:* $(1,-1)$*,* $(-3,1)$*,* $(0,1)$ *and* $(0:1:-1)$*.*

The minor relevant to the Macaulay resultant is the submatrix corresponding to the columns indexed by s^2, t^2 *and the rows indexed by* P_1, P_2*:* $\begin{vmatrix} 1 & 0 \\ 1 & -1 \end{vmatrix}$*.*

The Macaulay resultant can be adjusted to a homogeneous resultant without any difficulty: the homogeneous version is the original and the most robust. Indeed, the vanishing of the Macaulay resultant determines the existence of a common root in $\mathbb{P}^d(\mathbb{K})$.

2.2.2 Sparse resultants

The sparse resultants improves the Macaulay resultant in the sense that it also gives a determinental formula but uses a smaller matrix. The reduced matrix size comes from the fact that not all the parameters corresponding to monomials of smaller degrees than the polynomial degrees are required to construct a resultant but only a part of them. In other terms, it is a refinment of the polynomial space Δ.

Let $P(t_1, \dots, t_d) = \sum_{(k_1,\dots,k_d)\in\mathbb{N}^d} c_{k_1,\dots,k_d} t_1^{k_1} \dots t_d^{k_d}$ be a polynomial in d variables, with the coefficients $c_{k_1,\dots,k_d}$ being zero for almost every d-tuple. For convenience, we note $c_k := c_{k_1,\dots,k_d}$ the coefficients of P and we recall that $t^k := t_1^{k_1} \dots t_d^{k_d}$ is the multi-index power.

Definition 6. *The* support *of P is the set* $\mathsf{Supp}(P) \subset \mathbb{N}^d$ *of indices k such that $c_k \neq 0$.*

The Newton polytope *(or* Newton polygon *if $d = 2$) of P is the convex hull of its support:* $\mathsf{NPolytope}(P) := \mathsf{Hull}(\mathsf{Supp}(P))$.

The Newton polytope of a generic polynomial of degree δ is thus the simplex of dimension d and side-length δ. Newton polytopes give finer informations on polynomial supports than their degrees. It allows to create resultant formulae more fitting to sparse polynomials, that is polynomials with many zero coefficients (when looking at the coefficients corresponding to monomials of lower degrees than $\deg(P)$). When applied to non-sparse polynomials, the Newton polytopes are simplices and the sparse resultant matches with the Macaulay resultant.

We present two results using Newton polytopes: they can be used to compute the number of common roots of d polynomials in d variables and build a resultant matrix of $d + 1$ polynomials in d variables.

Number of common roots.
We first consider d polynomials $P_1, \dots, P_d$ in d variables and $\Delta_1, \dots, \Delta_d$ their Newton polytopes. Additionally, we make the assumption that these Newton polytopes are d-dimensional polytopes.

Definition 7. *The* Minkowski sum *of polytopes is* $\Delta_i + \Delta_j := \{a + b \mid a \in \Delta_i, b \in \Delta_j\}$.

The mixed volume *of* $(\Delta_i)_i$ *is given by:*

$$\mathsf{MV}(\Delta_1, \dots, \Delta_d) := \sum_{I\subset\{1,\dots,d\}} (-1)^{d-|I|}\, \mathsf{Vol}\left(\sum_{i\in I} \Delta_i\right) \tag{2.6}$$

Equivalently, MV *is the multilinear map w.r.t. the Minkowski sum and scalar multiplication such that* $\mathsf{MV}(\Delta_1, \Delta_1, \dots, \Delta_1) = d!\,\mathsf{Vol}(\Delta_1)$.

Example 4. *Let* $\begin{cases} P_0(s,t) = 1 + s + t + s^2 \\ P_1(s,t) = s + t + s^2\,t + s^2\,t^2 \\ P_2(s,t) = 1 + s + t + s\,t \end{cases}$ *. Then the Newton polygons of P_0, P_1 and P_2 and their Minkowski sum are the ones displayed in figure 2.2.*

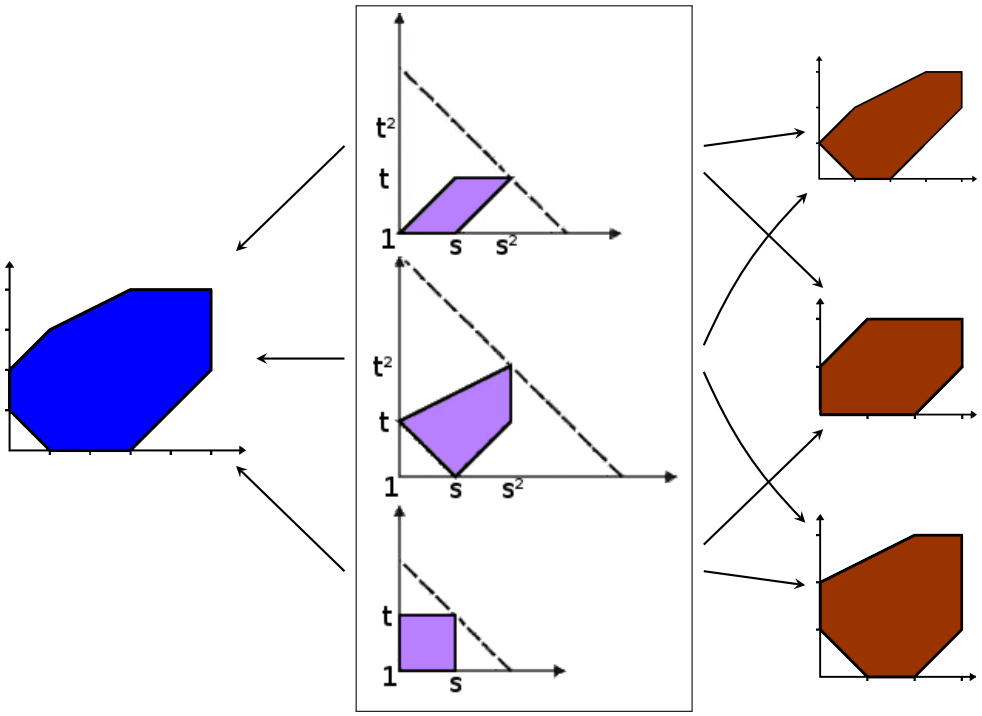


Figure 2.2: Newton polygons of the polynomials in the example 4 and their Minkowski sums

Theorem 4. [4] *The number of isolated common zeros (counted with multiplicity) of* d *polynomials in* $\left(\overline{\mathbb{K}}\backslash\{0\}\right)^d$ *is bounded by the mixed volume of their Newton polytopes:*

$$\#\,\text{RootsOf}(P_1,\ldots,P_d) \leq \text{MV}(\text{NPolytope}(P_1),\ldots,\text{NPolytope}(P_d)),$$
if $\dim(V(P_1,\ldots,P_d)) = 0$ *and* $t_j \nmid P_i,\ \forall i,j$

Moreover, this bound is exact for generic polynomials.

Note that if P_i is divisible by t_j, then the common roots can be split into RootsOf$(P_1,\ldots,P_i/t_j,\ldots,P_d)$ and RootsOf$(P_1|_{t_j=0},\ldots,P_{i-1}|_{t_j=0},P_{i+1}|_{t_j=0},\ldots,P_d|_{t_j=0})$.

Sparse resultant matrix.
We now consider $d+1$ polynomials $P_0,\ldots,P_d$ in d variables and $\Delta_0,\ldots,\Delta_d$ their Newton polytopes. Again, we assume that $\dim(\Delta_i) = d,\ \forall i$. We will need to compute $\text{MV}_i := \text{MV}(\Delta_0,\ldots,\Delta_{i-1},\Delta_{i+1},\ldots,\Delta_d)$ in order to build our matrix.

But first, because the formula 2.6 is not an efficient way to compute mixed volumes (much like Laplace's formula is not efficient when computing determinants of numerical matrices), we present an alternative computation method.

Consider $d+1$ generic affine functions $L_i : \mathbb{R}^d \to \mathbb{R}$ that we call lifting functions. Then $\hat{\Delta} = \sum_i \text{Hull}((p, L_i(p))_{p\in\Delta_i})$ is a convex polytope in $\mathbb{R}^{d+1}$. The lower hull of $\hat{\Delta}$, that is the faces $(\hat{F}_j)_j$ of $\hat{\Delta}$ such that $\forall j, \epsilon > 0, (F_k - (0,\ldots,0,\epsilon)) \cap \hat{\Delta} = \emptyset$, is projected to $\sum_i \Delta_i \subset \mathbb{R}^d$ and induces a subdivision of it into convex polytopes $\mathscr{F}$ with $F \subset \mathbb{R}^d,\ \forall F \in \mathscr{F}$.

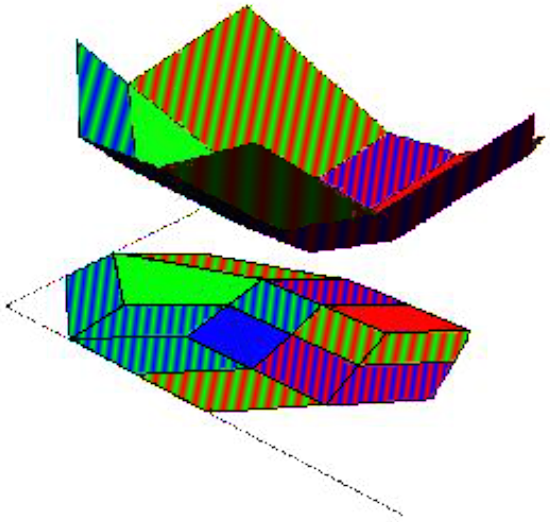


Figure 2.3: A subdivision obtained from the lower hull of the lifted Minkowski sum of the polynomials in the example 4

In this subdivision, the i-mixed cells, that is the polytopes $F \in \mathscr{F}$ such that $F = v_i + \left(\sum_{j\neq i} e_j\right)$ where v_i is a vertex of Δ_i and e_j are edges of $\Delta_j, \forall j$, can be used to compute the mixed volume:

Theorem 5. [24] *For generic lifting functions, we have:*

$$\mathsf{MV}_i = \mathsf{MV}(\Delta_0, \ldots, \Delta_{i-1}, \Delta_{i+1}, \ldots, \Delta_d) = \sum_F \mathsf{Vol}(F)$$

where the sum is taken over i-mixed cells F.

The sparse resultant matrix is constructed as follows: after constructing the subdivision above with generic lifting functions, pick a random vector $\vec{\epsilon} \in \mathbb{R}^d$ with $\|\vec{\epsilon}\|_\infty < 1$ and consider the points $\mathcal{E} := ((\sum_i \Delta_i) + \vec{\epsilon}) \cap \mathbb{N}^d$. All these points $p \in \mathcal{E}$ belong to the Minkowski sum of $(\Delta_i)_i$ and moreover, can be associated to an unique face $F_p \in \mathscr{F}$ such that $(p - \vec{\epsilon}) \in F_p$, provided that $\vec{\epsilon}$ is generic enough.

Then, define the following connection:

$$\begin{aligned} \mathsf{RC} : \mathcal{E} &\to \{0, \ldots, d\} \times \mathbb{N}^d \\ p &\mapsto (i, v_i) \text{ such that } \begin{cases} F_p = v_i + \left(\sum_{j\neq i} e_j\right) & \text{if } F_p \text{ is an } i\text{-mixed cell} \\ F_p = v_i + \left(\sum_{j\neq i} \lambda_j\right) & \text{otherwise} \end{cases} \end{aligned} \tag{2.7}$$

where v_i is a vertex of Δ_i, e_j are edges of Δ_j and λ_j are sub-faces of various dimensions of Δ_j. In the second case, there are several choices for i because F_p is the sum of several vertices and sub-faces of high dimensions; we pick the highest index i contributing to F_p with a vertex in that situation.

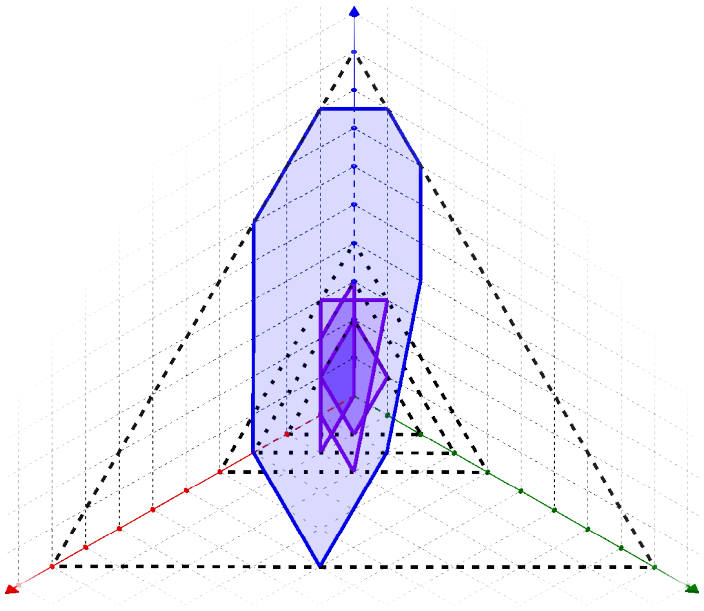


Figure 2.4: Newton polygons of the homogenized polynomials of the example 4 (*purple*) and their Minkowski sum (*blue*)

Now we can build a sparse resultant matrix with rows and columns indexed by $\mathcal{E}$:

$$M = (c_{i,q-p+v_i})_{p,q\in\mathcal{E}} \tag{2.8}$$

where $c_{i,k}$ is the coefficient of t^k of the polynomial P_i (or $c_{i,k} = 0$ if $k \notin \text{Supp}(P_i)$) and $\text{RC}(p) = (i, v_i)$. It is a square matrix of size equal or greater than $\sum_i \text{MV}_i$. It can be seen as the matrix of the map $(Q_0, \ldots, Q_d) \mapsto \sum_i Q_i P_i$ in the canonical basis of $\mathcal{E}$.

Theorem 6. [24] *Consider d-dimensional polytopes $\Delta_0, \ldots, \Delta_d \subset \mathbb{N}^d$ and let $\Delta := \{(P_0, \ldots, P_d) \mid \text{Supp}(P_i) = \Delta_i\}$.*

Then there is a resultant Res_Δ for polynomial collections $(P_0, \ldots, P_d) \in \Delta$. This resultant is of degree MV_i w.r.t. the coefficients of P_i. Moreover, the determinant of the matrix M constructed in (2.8) is a multiple of Res_Δ.

We developed a C++ implementation of this sparse matrix construction algorithm. It is available at `http://users.uoa.gr/~claroche/publications/SparseResultant.zip`.

Comparing the degrees, one can see that the determinant of M is equal to Res_Δ (up to a constant) when the cells F_p considered in the formula (2.7) are all mixed cells. That construction depending on both the lifting functions and the shifting vector $\vec{\epsilon}$, one can tweak them in hope of obtaining a smaller matrix at the end.

However, as for the Macaulay matrix construction, it is known that the extraneous factor can be expressed as a minor of the matrix M.

Finally, we say a few words about the projective versions of sparse resultants.

When applying the construction to homogenized polynomials, we obtain more symetric Minkowski sums as seen in the figure 2.4. The simplex obtained with dense polynomials

corresponds to a regular simplex. The condition of having polynomials non divisible by t_j is seen as having at least one point that belongs to the Newton polytope on each border of the regular simplex. Since the polytopes are not $d+1$-dimensional, the mixed volume formula of Theorem 5 does not apply, but a scaled formula can be used:

$$\mathsf{MV}_i = \sum_{F \in \{i\text{-mixed faces}\}} \frac{\mathsf{Vol}(F)}{\sqrt{d+1}}$$

However, the sparse resultant formula does not discriminate polynomials having common roots in $\mathbb{P}^d(\mathbb{K})$ but rather the polynomials having common roots in the toric space $(\mathbb{P}^1(\mathbb{K}))^d$ as shown in [14, Chapter 7, §3].

2.3 Interpolation matrices

This section describes a direct method to reduce implicitization to linear algebra by constructing a interpolation matrices $\overline{M}$ and $M(x)$, given a plane curve or a (hyper)surface in parametric form or as a point cloud. The matrix is indexed by all possible monomials in the implicit equation (columns) and different values (rows) at which all monomials get evaluated. The vector of coefficients of the implicit equation is in the kernel of these matrices, even in the presence of base points. This idea has been extensively used, e.g. [2, 21]. The matrix is somewhat different in [12], which is the method implemented in Maple for implicitization via the `algcurves[implicitize]` command. The latter method consists of expressing the implicit equations as the kernel of a carefully chosen integral form. It accepts non-algebraic parameterization and can still return formal implicit formulae provided that the integral form can be expressed with a formal formula (the integrand is polynomial in the parametric input equations). Alternatively, this method can perform floating-point computations and return approximate implicit formulae.

In [27], sparse elimination theory is employed to predict the implicit monomials and build the interpolation matrix. The monomial set is determined quite tightly for parametric models, by means of the sparse resultant of the parametric polynomials, thus exploiting the sparseness of the parametric and implicit polynomials.

More specifically, if the input object is affinely parameterized by rational functions as in the equations (3) then it is possible to predict the implicit monomials. This set is included in the predicted (implicit) polytope computed by software `ResPol`[25]. If the input is a point cloud, we consider a coarse estimation of the monomial set by guessing the total degree of an implicit representation and taking all the monomials of that degree or lower. Let S be the predicted set of implicit monomials and $|S|$ its cardinality.

The set S is used to construct a numerical matrix $\overline{M}$, expressing a linear system whose unknowns are the coefficients c_i ($i = 1, \ldots, |S|$) of the monomials S in the implicit polynomial, as discussed above. If the input object is a parameterization, we obtain the linear system in the c_i by substituting each x_j by its rational parametric expression $x_j = \frac{f_j(t_1,\ldots,t_d)}{f_0(t_1,\ldots,t_d)}$ in the equation $\sum_{i=1}^{|S|} c_i x^{a_i} = 0$, where $x^{a_i} := x_1^{a_{i1}} \cdots x_n^{a_{in}}$. We then evaluate the parameters

$t = (t_1, \ldots, t_d)$ at generic points (randomized in practice) $\tau_k \in \mathbb{C}^{n-1},\ k = 1, \ldots, \mu,\ \mu \geqslant |S|$, avoiding values that make the denominators of the parametric expressions close to 0. Each evaluation gives a linear equation in the coefficients c_i.

Letting $m_i = m_i(t)$ denote the monomial x^{a_i} after substituting each x_j by its parametric expression in the equations 3, and $m_i|_{t=\tau_k}$ its evaluation at $t = \tau_k$, we end up with a matrix $\overline{M}$ of the form:

$$\overline{M} = \begin{pmatrix} m_1|_{t=\tau_1} & \cdots & m_{|S|}|_{t=\tau_1} \\ \vdots & \ddots & \vdots \\ m_1|_{t=\tau_\mu} & \cdots & m_{|S|}|_{t=\tau_\mu} \end{pmatrix}.$$

Typically $\mu = |S|$ for performing exact kernel computation, and $\mu = 2|S|$ for approximate numeric computation.

If the input object is given as a point cloud, we take μ random points out of it ($\mu \geqslant |S|$) and use them instead of the points evaluated at the parameters $\tau_k,\ k = 1, \ldots, \mu$.

Let M' be the $(|S| - 1) \times |S|$ numeric matrix obtained by evaluating the monomials S at $|S| - 1$ points $\tau_k,\ k = 1 \ldots, |S| - 1$. We obtain the $|S| \times |S|$ matrix $M(x)$, which is numeric except for its last row, by appending the row of monomials S to matrix M':

$$M(x) = \begin{pmatrix} M' \\ S(x) \end{pmatrix}, \tag{2.9}$$

where we use the notation $S(x)$ to emphasize that this is the only symbolic row of $M(x)$. Notice that matrices $\overline{M}$, M' and $M(p)$, for a point p lying on the hypersurface, have the same kernel. Matrix $M(x)$ has an important property:

Lemma 7. [26, Lemma 7] *Assuming M' is of full rank, then* $\det M(x)$ *equals the implicit polynomial up to a constant.*

We now generalise this interpolation matrix construction to the case of varieties of codimension greater than 1.

Let $V \subset \mathbb{K}^n$ be a variety of any codimension given parametrically or as a point cloud. Given a set of monomials S, we randomly pick μ ($\mu \geqslant |S|$) points $\overline{x}_k \in V,\ k = 1, \ldots, \mu$ on V either from the input point cloud, or as evaluations $\overline{x}_k = \left(\frac{f_1(\tau_k)}{f_0(\tau_k)}, \ldots, \frac{f_n(\tau_k)}{f_0(\tau_k)}\right)$ of the input parametric equations at random points τ_k. Then we construct the interpolation matrix $M(x) = \begin{pmatrix} M' \\ S(x) \end{pmatrix}$ as previously, where M' is a numeric submatrix, $x = (x_1, \ldots, x_n)$ are symbolic coordinates and $S(x)$ is a row of symbolic monomials in x.

Recall that the support of a polynomial is the set of powers of the monomials appearing with non-zero coefficient. We write $\mathsf{Supp}(S) := \cup_{m \in S} \mathsf{Supp}(m)$ and define the following set of polynomials:

$$\mathcal{P} \coloneqq \{P \in \mathbb{C}[x_1, \ldots, x_n] \mid \mathsf{Supp}(P) \subset \mathsf{Supp}(S) \text{ and } \forall\, \xi \in V,\ P(\xi) = 0\}.$$

$\mathcal{P}$ is a $\mathbb{C}$-vector space. We assume that S contains all the monomials of a set of generators of the ideal $\mathcal{I}(V)$, i.e. $V = \{\xi \in \mathbb{C}^n \mid \forall P \in \mathcal{P},\, P(\xi) = 0\}$. This construction is summarized in the algorithm A.1.

Suppose also that the points $\overline{x}_k$, $k = 1, \ldots, \mu$, defining the rows of M' are chosen generically enough, so that for all $P \in \mathbb{C}[x]$ with monomials in S, we have:

$$P(\overline{x}_k) = 0,\, \forall k \in \{1, \ldots, \mu\} \iff P \in \mathcal{P}$$

Then, the matrix $M(x)$ has a drop-of-rank property but weaker than that of Lemma 7 in the sense that M' is not of full rank.

Lemma 8. [29, Lemma 4] *Assume we built $M(x)$ using a set S that contains all the monomials of a set of generators of the ideal $\mathcal{I}(V)$. Then, for $\xi \in \mathbb{C}^n$, ξ belongs to V if and only if* $\text{rank}(M(\xi)) < \text{rank}(M(x))$*, where it holds* $\text{rank}(M(x)) = \text{rank}(M') + 1$.

Proof. Using the basis S of monomials, we consider the canonical complex space of dimension $|S|$, $\mathbb{C}^{|S|}$. It is isomorphic to the space of polynomials $\{P \in \mathbb{C}[x] \mid \text{Supp}(p) \subset \text{Supp}(S)\}$.

By abuse of notation, the image of $\mathcal{P}$ under this isomorphism will also be called $\mathcal{P}$. By the hypothesis on genericity of $\overline{x}_k$, $k = 1, \ldots, \mu$, we have that $\text{Ker}(M') = \mathcal{P}$. So, for $\xi \in \mathbb{C}^n$, we have:

$$\begin{aligned}
&\xi \in V \\
&\iff \forall P \in \mathcal{P},\, P(\xi) = 0 \text{ (by the hypothesis that } \mathcal{P} \text{ characterizes } V) \\
&\iff \text{Ker}(M(\xi)) = \mathcal{P} = \text{Ker}(M') \\
&\iff \text{rank}(M(\xi)) = \text{rank}(M') < \text{rank}(M(x)).
\end{aligned}$$

□

In practice, taking $\mu = |S|$ and random points $\overline{x}_k$, $k = 1, \ldots, \mu$, is enough to satisfy the hypothesis of Lemma 8. The set of monomials S is hard to determine optimally. One can estimate bounds on the degree of implicit representations of V and take all monomials of degree up to that bound, which usually leads to more monomials and a larger matrix $M(x)$ than needed. If the input is a rational parameterization $\left(\frac{f_1}{f_0}, \ldots, \frac{f_n}{f_0}\right)$ of V, we have an upper bound of $\deg(V)$ given by $\deg(V) \leqslant \prod_{i=1}^{d} \max_j(\deg_{t_i} f_j)$ where $\deg_{t_i} f_j$ is the degree of f_j in the i-th parameter.

The drop-of-rank property readily leads to a computation of the implicit equations representing V set-theoretically, either by computing all the maximal non-zero minors of $M(x)$ containing the last line, which is inefficient in practice, or by computing the nullspace of M'.

The matrix representations given in Chapter 5 also have drop-of-rank properties but they are much smaller. The construction of those matrices, unlike $M(x)$ presented here, relies on syzygy computations and is thus slower. However, those methods are overall more efficient because of the faster rank computation at each point evaluation.

3. SWEPT VOLUMES

In this section, we introduce the notion of *rigid transformations* and *swept volumes*. These two notions are used in CAGD for designing 3D objects through boolean operations. Considering a shaping tool $\mathcal{B}$, like a drilling or milling machine, progressively removing parts of a 3D object O, the movement of the tool will follow a time-dependent rigid transformation $\mathcal{T}(t)$ assuming that it cannot be deformed. The result of this operation is a shaped 3D object O' that is the difference of the base object by the tool swept along that rigid transformation:

$$O' = O \backslash \mathcal{T}(\mathcal{B}), \text{ where } \mathcal{T}(\mathcal{B}) = \cup_t [\mathcal{T}(t)](\mathcal{B})$$

The goal of this section is to give an efficient implicit representation of the swept volume $\mathcal{T}(\mathcal{B})$. This implicit representation is then used to perform the above boolean difference with the object to be shaped.

In the following, the shaping tool $\mathcal{B}$ is called the *base volume* in opposition to the swept volume $\mathcal{T}(\mathcal{B})$.

Starting with a point cloud of the base volume, we build a data structure enabling this kind of operations:

- Given a point $P \in \mathbb{R}^3$, does the point P belong to the swept volume $\mathcal{T}(\mathcal{B})$?
- What is the distance between $P \in \mathbb{R}^3$ and $\mathcal{T}(\mathcal{B})$?
- Given a ray R, what is the first intersection of R with $\mathcal{T}(\mathcal{B})$? What are all its intersections?
- Given an other object O, what is the boolean subtraction $O \backslash \mathcal{T}(\mathcal{B})$?

One way to proceed is first generate a point cloud of the swept volume and then implicitize that point cloud (depicted in red in Fig. 3.1). Here, however, we take a different path: we first implicitize the base volume and only then we use the transformations to build an implicit representation of the swept volume (depicted in grey in Fig. 3.1). This way, we can build an implicit representation that fits to the swept feature of $\mathcal{T}(\mathcal{B})$, allowing more details in its geometry due to the fact that the details of the base volume is carried to the details of the swept volume.

Constructing a data structure suited for swept volumes not only allows to perform implicit operations but also give more specific answers such as: if a point P belongs to the swept volume, for which t is it inside $[\mathcal{T}(t)](\mathcal{B})$? which part(s) of the base volume meet with P? etc.

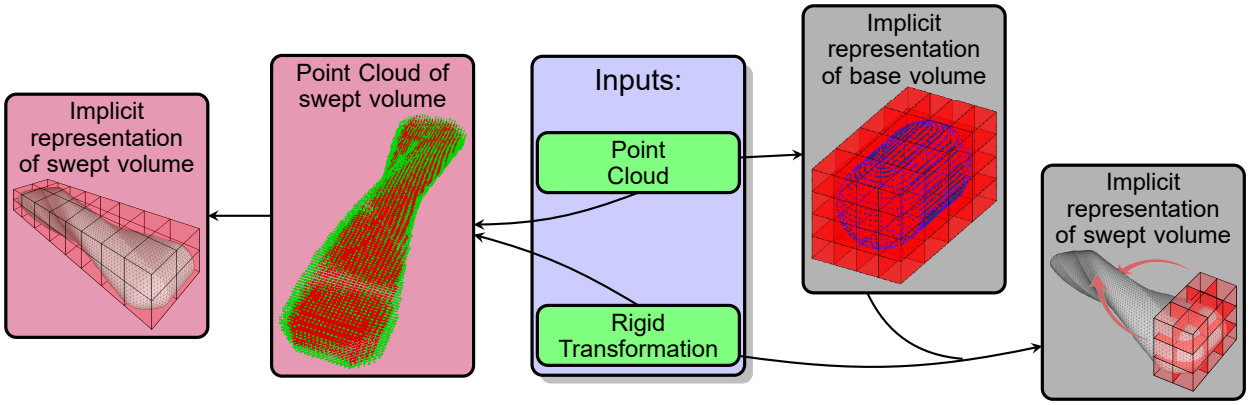


Figure 3.1: Two possible representation strategies: construct a data structure of the swept point cloud (*left*), construct a data structure using both the point cloud of the base volume and transformation informations (*right*)

3.1 Implicitizing a point cloud

Definition 8. *A* local implicit representation *of a base volume $\mathcal{B}$ is a collection $(A_i, F_i)_{1\leq i\leq N}$ of bounded areas A_i (cubes, balls, ...) and of implicit procedures $F_i : A_i \to \mathbb{R}$. The 3D model $\mathcal{B}$ is then given by $\mathcal{B} = \{(x, y, z) \mid (x, y, z) \in A_i$ and $F_i(x, y, z) \leq 0\}$.*

We first describe two different ways of constructing a local implicit representation of a base volume from a given point cloud. The two algorithms presented here need a point cloud with both point coordinates and normals.

In the following, $\mathcal{P}$ and $\mathcal{N}$ are a given 3D point cloud of an object's surface and the outer normals of these points, respectively.

3.1.1 MPU method

The Multi-level Partition of Unity implicitization[52], or MPU, is an algorithm generating an octree-based local implicit representation. In other terms, the areas A_i are cuboids whose edges are parallel to the axes. A local approximation procedure F_i can be of three types in order to adapt to the local shape of $\mathcal{B}$:

(a) a general 3D quadratic polynomial,

(b) a bivariate quadratic polynomial in local coordinates,

(c) a piecewise quadratic polynomial for representing edges and corners (2, 3 or 4 pieces depending on the situation).

At each step of the algorithm, we subdivide the cuboids inside which the local approximation are not precise enough into 8 smaller cuboids. Then we update the local approximations inside these 8 smaller cuboids. In order to increase the representation's smoothness, the local approximation inside a cuboid is computed by taking into account all the points inside an ellipsoid containing the cuboid. The precision of a local approximation is computed using the Taubin distance (see [68]).

When computing local approximations of type (a), we first generate a small pointset Q that can be used to obtain a reliable estimate of a signed distance function. We then compute the quadratic polynomial f that minimizes the following quantity:

$$\frac{1}{\sum_i w(p_i)} \sum_i w(p_i) f(p_i)^2 + \frac{1}{|Q|} \sum_{q \in Q} (f(q) - d)^2 \tag{3.1}$$

where w, d and Q are defined in the algorithm A.3.

In the case of local approximations of type (b), a local coordinate system (u, v, n) is introduced, where n is a weighted arithmetic mean of the point cloud's normals. A bivariate quadratic polynomial f in (u, v, n) is then a polynomial of the form:

$$f(p) = w - \left(c_{20}u^2 + c_{11}uv + c_{02}v^2 + c_{10}u + c_{01}v + c_{00}\right) \tag{3.2}$$

where c_{ij} are the polynomial's parameters and u, v, w are the coordinates of the point p in the coordinate system (u, v, n).

In the case (c) of a sharp feature, we compare the different normals in order to determine whether there is an edge, a three-sided corner or a four-sided corner (see [43]). Then, we split the points into two, three or four pointsets respectively and compute local approximations f_k of types (b) on each of these pointsets separately. The local implicit procedure is then given by $f(p) = \min_k f_k(p)$.

The algorithm A.2 sketches the main loop of MPU while the algorithm A.3 details the computation of the different types of local approximations.

3.1.2 Slim method

The Sparse low-degree implicitization[53], or Slim, is an algorithm generating an ball-based local implicit representation. In other terms, the areas A_i are balls and intersections of balls. The local approximation procedures F_i are bivariate quadratic polynomials in local coordinates, much like the procedures of type (b) of the MPU method. We can use a more restricted variety of local approximation procedures because we have a better control over the positioning of the areas. Indeed, while the MPU areas are all cuboids (or cubes in the rescaled space) partitioning the object's bounding box, here we use spheres that we can centre on the object's surface, with no fear of having remote areas containing only a small portion of the object in its corner.

The drawback is the need for overlapping spheres in order to cover the whole object. As polynomial continuity can hardly be satisfied in the overlapping areas, and certainly not

with low-degree polynomials, another approach is used: in these areas, the polynomials are weighted depending on the point's distances to the centres of the overlapping balls. Of course, the weights are computed on-the-fly when the ownership of a query point is asked (or the intersection of the object with a query ray must be computed): only the local quadratic polynomials tied to single balls are stored in the representation.

Also, the query points that are not covered by the spheres may be inside or outside the object. When asking the ownership of a query point q in this situation, simply search for its nearest neighbour p in $\mathcal{P}$ and check the sign of $< q-p, n >$ where n is p's outer normal. When negative, q is inside the object with a signed distance close to $-\|p-q\|$. When positive, q is outside the object with a signed distance close to $\|p-q\|$.

Slim uses compactly supported Gaussian-like weights:

$$G_R(r) := \begin{cases} \exp\left(-\frac{1}{1-(r/R)^2}\right) & \text{if } r \in (-R, R) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

We first need to cover $\mathcal{P}$ by a set of balls of a given radius. A simple and efficient way to do it is to pick a random point from $\mathcal{P}$ as the centre of the first ball and then continue picking random points as the centres of the subsequent balls amongst those that are not yet covered. This way, the centres of all the balls used in the algorithm are points of $\mathcal{P}$.

Given a ball $B = B(c, R)$, a rough estimation of the surface's normal n close to c is obtained as the average of the normals of $\mathcal{P} \cap B$. A local coordinate system (u, v, n) centred on c is used: quadratic polynomials in this local system are of the form (3.2). The best local approximation w.r.t. the ball B is then the quadratic polynomial F_B minimizing the following quantity:

$$\sum_{p \in \mathcal{P} \cap B(c,R)} G_R(\|p-c\|) F_B(p)^2 \tag{3.4}$$

Once a local approximation F_B is computed, two rankings are assigned to a radius ρ:

$$\begin{aligned} \epsilon(\rho) &:= \sum_{p \in \mathcal{P} \cap B(c,\rho)} F_B(p)^2 \\ E(\rho) &:= \epsilon(\rho) + \lambda (T_{\mathsf{MDL}}/\rho)^2 \end{aligned} \tag{3.5}$$

where T_{MDL} is a parameter and λ is a regularizing constant computed once: it is set as the average of the minimum eigenvalues of the co-variance matrices of each point $p \in \mathcal{P}$ with its ten nearest neighbours in $\mathcal{P} \backslash \{p\}$.

With this, the Slim algorithm consists of the computations of local approximations w.r.t. balls of gradually smaller radius ρ_k and stop when the quantities $E(\rho_k)$ attains a suitable local minimum. The balls and local approximations computed at each step can be kept in order to have a multi-scale approximation: if only a rough approximation is required for a specific query, we can use the few big balls of early steps instead of the many small balls of late steps.

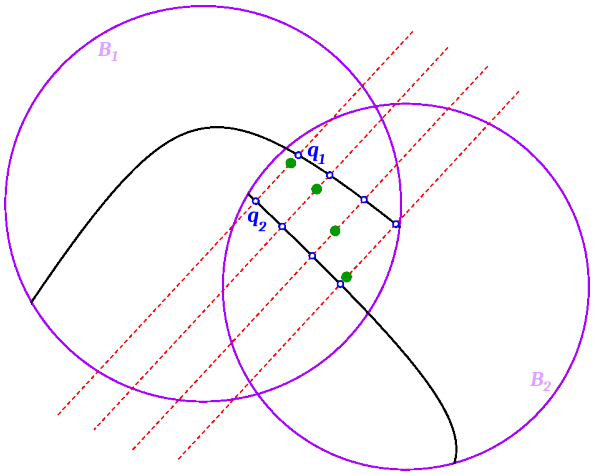


Figure 3.2: Slim: ray intersection in overlapping spheres.

Once the representation structure is generated (see A.4), the only thing left is how overlapping areas must be dealt with. Consider a query point $q \in B_1 \cap \cdots \cap B_m$ where B_j are balls of centre c_j and radius r_j given by a Slim representation. Then $F_{B_1 \cap \cdots \cap B_m}(q) := \dfrac{\sum_j G_{r_j}(\|q - c_j\|) F_{B_j}(q)}{\sum_j G_{r_j}(\|q - c_j\|)}$. The query point q belongs to the object iff $F_{B_1 \cap \cdots \cap B_m}(q) \leq 0$. That way, the transitions of the surfaces between the balls B_j are smoothened.

Similarly, the intersection of the object with a query ray ℓ is given by $\dfrac{\sum_j G_{r_j}(\|q_j - c_j\|) q_j}{\sum_j G_{r_j}(\|q_j - c_j\|)}$ where q_j are the intersections of ℓ with the local surfaces given by F_{B_j} and the balls B_j taken into account are only the first ones:

$\{B_j\}_j :=$ {ball B of the Slim representation such that $\ell \cap B_1 \cap B \neq \emptyset$, where B_1 is the first ball intersected by ℓ} (see Fig. 3.2).

3.2 Swept volume data structure

Definition 9. *A* rigid transformation T *is a map*

$$\begin{array}{rccl} \mathcal{T}: & [a,b] & \to & \mathsf{Iso}(\mathbb{R}^3) \\ & t & \mapsto & \mathsf{Transl}_{v(t)} \circ \mathsf{Rot}_{\alpha(t),\beta(t),\gamma(t)} \end{array}$$

where $v : [a,b] \to \mathbb{R}^3$ *and* α, β, γ *are piecewise polynomials and* $\mathsf{Transl}_v, \mathsf{Rot}_{\alpha,\beta,\gamma}$ *are respectively the translation of vector* v *and the rotation of Euler angles* (α, β, γ).

A swept volume $\mathcal{T}(\mathcal{B})$ *of base* $\mathcal{B}$ *and of rigid transformation* $\mathcal{T}$ *is* $\mathcal{T}(\mathcal{B}) := \cup_{t \in [a,b]} [\mathcal{T}(t)](\mathcal{B})$.

Example 5. *Let* $\mathcal{B}$ *be a capsule-like shape:*

$$\begin{aligned} \mathcal{B} = ((&B((-2,0,0), \sqrt{2}), y^2 + z^2 - x - 2), \\ &(B((0,0,0), \sqrt{2}), y^2 + z^2 - 1), \\ &(B((2,0,0), \sqrt{2}), y^2 + z^2 + x - 2)) \end{aligned}$$ *where* $B(x,r)$ *is the ball of centre* x *and radius* r

And $\mathcal{T}$ a linear interpolation between Id *and* $\mathsf{Transl}_{(0,16,0)} \circ \mathsf{Rot}_{0,\pi,0}$*:*

$$[\mathcal{T}(t)](x,y,z) = \begin{pmatrix} \cos(\pi t) & 0 & -\sin(\pi t) \\ 0 & 1 & 0 \\ \sin(\pi t) & 0 & \cos(\pi t) \end{pmatrix} . \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} 0 \\ 16t \\ 0 \end{pmatrix}, \text{ for } t \in [0,1]$$

This swept capsule-like shape is drawn in figure 3.1.

We describe how, given $\mathcal{B}$ and $\mathcal{T}$, we construct a local implicit representation of $\mathcal{T}(\mathcal{B})$.

In previous works, such implicit representation of swept volumes have been developed for specific types of base volumes. For instance, the boundary of swept volumes of convex polyhedrons are ruled surfaces; that property is used for the implicitization algorithms described in [42, 72]. As swept volumes have many applications in robotics and collision detection, another algorithm described in [69] handles base volumes made of shifted convex polyhedrons (i.e. points at a given "safety" distance of a convex polyhedron). Also, in [56], swept cuboids are approximated for the purpose of a real-time planning of a walking robot's movements. In the following, though, we assume that the base volume can be anything in the range of the definition 8.

Let $\mathbb{B}$ be a bounding box of $\mathcal{T}(\mathcal{B})$. We split $\mathbb{B}$ into cells $(C_j)_{1 \leq j \leq M}$ and compute $\mathcal{A}_j := \{(A_i, [t_0, t_1]) \mid \forall t \in [t_0, t_1], C_j \cap [\mathcal{T}(t)](A_i) \neq \emptyset\}$. It is the list of local areas of $\mathcal{B}$ intersecting the cell C_j along the swept transformation and the times between which they intersect (see the figure 3.3. How we split $\mathbb{B}$ into cells and how we compute $(\mathcal{A}_j)_j$ in practice is explained further.

Given a swept volume, we choose a suitable partition $(C_j)_j$ and compute $(\mathcal{A}_j)_j$ once. The tree structure given by $(C_j, \mathcal{A}_j)_j$ is our preprocessing structure. It allows to filter the relevant areas used for checking whether a point belongs to the swept volume or not. Proceeding that way, the local procedures F_i are not requested at all at the preprocessing step: only the intersection of relatively simple objects, the moving areas (moving spheres, moving cuboids,…) and the cells (rectangular cuboids), must be computed.

Once the tree structure is known, let $P \in \mathbb{R}^3$ be a query point. If $P \notin \mathbb{B}$, we return that $P \notin \mathcal{T}(\mathcal{B})$. Else, using the preprocessing structure, we find j such that $P \in C_j$ in $O(\log(M))$ time complexity. We then perform more accurate checks on P, using a numerical solver to find

$$\min \left\{ F_i([\mathcal{T}(t)^{-1}](P)) \mid A_i \text{ and } t \text{ are in } \mathcal{A}_j \right\}$$

This can be performed in $O(|\mathcal{A}_j| \log(\epsilon^{-1})\tau_j)$ worst-time complexity using the bisection algorithm, where ϵ is the solver precision and τ_j the size of the time segments $[t_0, t_1]$ in $\mathcal{A}_j$. It can be performed faster if the hypotheses on F_i allow better algorithms to be used (typically, the Newton method when one can compute the differential of F_i).

Thus, we want $|\mathcal{A}_j|$ and τ_j to be rather small. We are interested in computing a partition of $\mathbb{B}$ by cells $(C_j)_j$ minimizing the following quantity:

$$\mathsf{Cost}((C_j)_j) := \log(M) + \frac{1}{M} \sum_{j=1}^{M} \mathsf{Vol}(C_j) |\mathcal{A}_j| \tau_j \tag{3.6}$$

The use of a mean measure weighted by the size of cells instead of the maximal value is motivated by the objective to give an implicit procedure that would likely be used on a lot of points. One can add more sophisticated weights if parts of the models are more likely to be processed than others (for instance, if there is a visible face of the swept volume and a back face that is not usually rendered). Such a weight can be introduced by considering $\mathrm{Cost}_\omega((C_j)_j) := \log(M) + \frac{1}{\int_{x\in\mathbb{B}}\omega(x)\,\mathrm{d}x}\sum_{j=1}^{M}\left(\int_{x\in C_j}\omega(x)\,\mathrm{d}x|\mathcal{A}_j|\tau_j\right)$ where $\omega : \mathbb{B} \to \mathbb{R}_+$ is a bounded user-specified weight that is high-valued in the important areas of the swept volume and low-valued in less important areas.

In order to minimize the cost, we split the bounding box of the swept volume, $\mathbb{B}$, according to the following procedure:

1. Start with a trivial partition $C_1 := \mathbb{B}$.
2. Pick many parameters $(t_k)_k$ in $[a, b]$ and consider the rigid transformation at time t_k applied to the local areas, $S_{i,k} := [\mathcal{T}(t_k)](A_i)$.
3. While the cost of the partition (3.6) decreases, pick the cell with the largest (weighted) volume and split it along a coordinate in two other cells c_1, c_2 by optimizing $\#\{(i,k) \mid S_{i,k} \cap c_1 \neq \emptyset\} + \#\{(i,k) \mid S_{i,k} \cap c_2 \neq \emptyset\}$.
4. For the cells of the boundary, find the best split that would generate an emply cell (i.e. with no intersection with $\cup_{i,k} S_{i,k}$). If that empty cell has a surface large enough (possibly weighted by ω), then perform the split.

Now, we develop the way to compute $\mathcal{A}_j$, the local areas intersecting the cell C_j. This step relies heavily on the basic shapes used for the local areas A_i; the method must be adapted depending on what shape is used. Since the cells C_j themselves are rectangular cuboids, the computation of $\mathcal{A}_j$ consists of solving rectangular cuboid/rectangular cuboid intersection problems (when $\mathcal{B}$ was generated by MPU) or sphere/rectangular cuboid intersection problems (when $\mathcal{B}$ was generated by Slim) etc., one of which being moving (i.e. depending on a parameter t). Either A_i or C_j can be chosen to depend on the time parameter; this choice corresponds to solving either one of the two equivalent problems:

$$\text{Solve } [\mathcal{T}(t)](A_i) \cap C_j \neq \emptyset \text{ w.r.t. } t, \tag{3.7}$$
$$\text{Solve } [\mathcal{T}(t)^{-1}](C_j) \cap A_i \neq \emptyset \text{ w.r.t. } t. \tag{3.8}$$

When A_i is a sphere, it is more efficient to use the first alternative since it means applying $\mathcal{T}(t)$ less times (we apply it only on the centre of the sphere, instead of applying it to each of the 6 cuboid's faces). When A_i is a more complicated shape than C_j, we use the second alternative instead. That is what we do when deciding whether a point P belongs to $\mathcal{T}(\mathcal{B})$: we compute parametrically the ownership of $[\mathcal{T}(t)^{-1}](P)$ to $\mathcal{B}$ instead of the ownership of P to $[\mathcal{T}(t)](\mathcal{B})$.

Let $f_{k,-1}, f_{k,+1}$ (with $k \in \{1,2,3\}$) be the normalised equations of the 6 faces of C_j. For ease of notations, we will use k, k' and k'' such that $\{k, k', k''\} = \{1,2,3\}$ so that each one

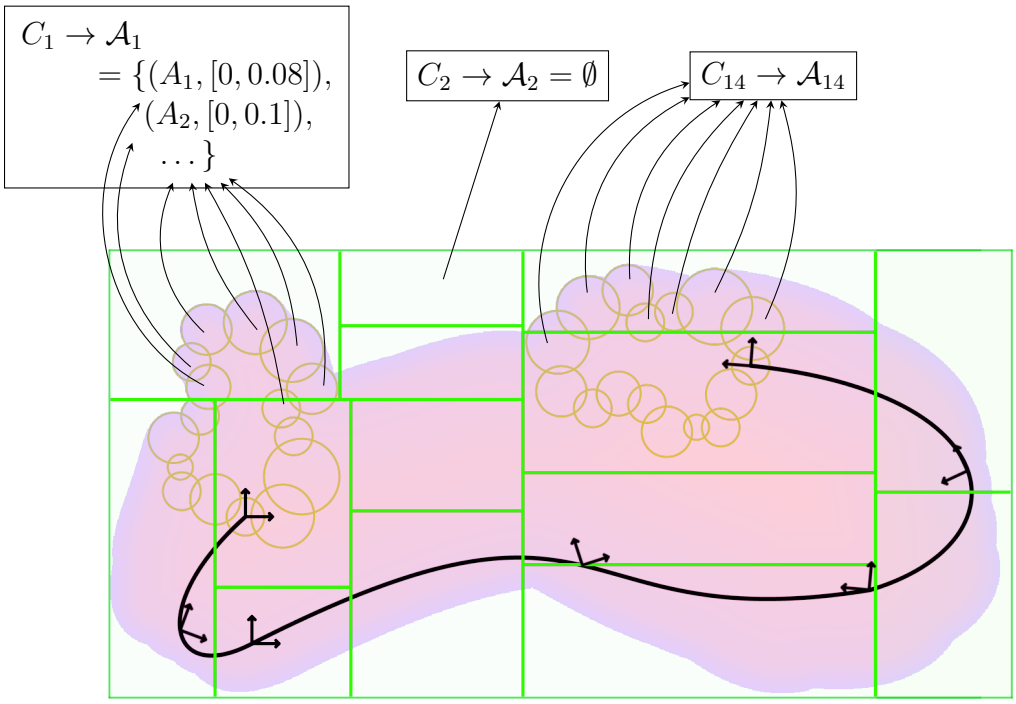


Figure 3.3: Tree structure $(C_j, \mathcal{A}_j)_j$ in 2D with circles as local areas (*yellow*). The rigid transformation (*black* curve with orientation) is applied on the local areas (*purple* surface) and used to construct the cells (*green*): each one of these cells is associated with the part of $\mathcal{B}$ and the time span that are relevant. Note that the local implicit procedures F_i are not involved at this step.

corresponds to one coordinate. By *normalised equations* of faces, we mean equations of the form $f_{k,\pm1}(P) = P.\vec{n} - d$ where $\vec{n}$ is the unit outward-pointing normal and d is a suitable constant ($d = Q.\vec{n}$ for a point Q of the face). This way, $f_{k,\pm1}$ are the signed-distance functions of the faces of C_j. Notice that, since the cell C_j has the same orientation as the axes, $f_{k,\pm1}$ actually depends only on one coordinate. Also, let $e_{k,\sigma_1,k',\sigma_2}$ (with $\sigma_k \in \{-1, +1\}$) be the edge of C_j defined by $f_{k,\sigma_1} = f_{k',\sigma_2} = 0$ and $v_{\sigma_1,\sigma_2,\sigma_3}$ be the vertex defined by $f_{1,\sigma_1} = f_{2,\sigma_2} = f_{3,\sigma_3} = 0$.

Now, suppose that A_i is a sphere of centre O and radius R. The moving centre $[\mathcal{T}(t)](O)$ is thus given by $O(t) := M(t).O + v(t)$ where $M(t)$ is the rotation matrix of Euler angles $(\alpha(t), \beta(t), \gamma(t))$. The problem (3.7) can then be described by the following equations:

$$\begin{aligned}
&f_{k,\sigma_1}(O(t)) - R_i = 0 \text{ and } f_{k',\pm1}(O(t)) \leq 0 \text{ and } f_{k'',\pm1}(O(t)) \leq 0 \\
\text{or } &\text{Dist}(O(t), e_{k,\sigma_1,k',\sigma_2})^2 - R^2 = 0 \text{ and } f_{k,\sigma_1}(O(t)) > 0 \text{ and } f_{k',\sigma_2}(O(t)) > 0 \text{ and } f_{k'',\pm1}(O(t)) \leq 0 \\
\text{or } &\text{Dist}(O(t), v_{\sigma_1,\sigma_2,\sigma_3})^2 - R^2 = 0 \text{ and } f_{1,\sigma_1}(O(t)) > 0 \text{ and } f_{2,\sigma_2}(O(t)) > 0 \text{ and } f_{3,\sigma_3}(O(t)) > 0
\end{aligned}$$

which makes 6 equations to solve for the first case, plus 12 for the second case and 8 for the third case for a total of 26 equations per sphere/cuboid couples. For all the solutions found, several inequalities must be checked but these are not expensive.

Remark 1. *Note that it is possible to approximate the structure $\mathcal{A}_j$ by solving $f_{k,\sigma_1}(O(t)) - R_i = 0$ instead. When doing that, there are only 6 equations to solve per sphere/cuboid*

couples, which effectively makes the preprocessing computation faster at the price of a slightly slower runtime for membership checks and ray intersections.

An other way to speed up this preprocessing step, notice that if an area A_i is in contact with a cell C_j for $t \in [t_0, t_1]$, then the $A_{i'}$ cannot be in contact with any cell $C_{j'}$ such that $\text{Dist}(C_j, C_{j'}) + \text{Diameter}(A_i) + \text{Diameter}(A_{i'}) > \text{Dist}(A_i, A_{i'})$ *in the same time period. Thus, using the informations on the already computed area positions allows to filter out a few cells when processing the areas that are nearby the former one.*

The algorithm A.5 sketches how an implicit representation of $\mathcal{T}(\mathcal{B})$ is computed and the algorithm A.6 shows how to use that implicit representation, both as an ownership oracle and as a ray intersection test.

4. CHOW FORMS

One of the implicitization methods developed is inspired by the theory of Chow forms.

4.1 Chow variety

4.1.1 A hypersurface of the Grassmannian space

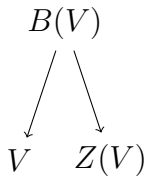
Chow forms have been studied in computer algebra, in particular for varieties of codimension $\geqslant 2$, since they provide a method to describe the variety by a single polynomial [17, 31]. The Chow form of a variety V is basically a polynomial R_V which indicates when linear subspaces intersect V. For example, the Chow form of a space curve in projective 3-dimensional space is a polynomial in the indeterminates u_{ij} that vanishes whenever the planes

$$\begin{aligned} H_0 &= u_{00}x_0 + u_{01}x_1 + u_{02}x_2 + u_{03}x_3 = 0, \\ H_1 &= u_{10}x_0 + u_{11}x_1 + u_{12}x_2 + u_{13}x_3 = 0, \end{aligned} \tag{4.1}$$

intersect on the curve. If the space curve is given parametrically, the Chow form represents the variety in terms of R_V. It can be computed by a *symbolic resultant* of the system of linear equations (4.1) where the set of variables $X = (x_i)_i$ is substituted with the parametric equations; the resultant eliminates the parameters and yields a polynomial in the variables $U = (u_i)_i$. The implicit hypersurfaces in X containing the variety have to be extracted through *rewriting rules*. These make implicitization algorithms that rely on the computation of R_V impractical for varieties of high degree and/or dimension.

Due to their complexity, very few implementations exist for computing the Chow forms themselves. Amongst them, [65] is an implementation in Macaulay2, based on the formula of the Chow form in the Grassmannian space using the Plücker coordinates, and [37, Subroutine 7] is an algorithm using polynomial ring tools and based on a Poisson-like formula of the Chow form.

To formally define the Chow form let $Gr(k+1, n+1)$ denote the Grassmannian space of k-dimensional linear projective subspaces of $\mathbb{P}^n$. For a variety $V \subset \mathbb{P}^n$ of codimension c, let $B(V) \subset \mathbb{P}^n \times Gr(c, n+1)$ be the set of (P, L) such that P belongs both to V and to the projective linear subspace L of dimension $c-1$. Then we obtain V by forgetting the second component in $B(V)$ and we obtain an hypersurface $Z(V) \coloneqq \{L \in Gr(c, n+1) \mid L \cap V \neq \emptyset\}$ of the Grassmannian space $Gr(c, n+1)$ by forgetting the first component in $B(V)$.



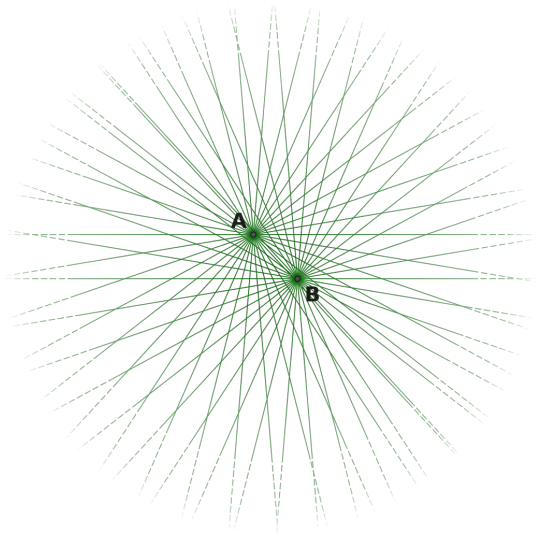


Figure 4.1: Chow form of a zero-dimensional variety $V = \{A, B\}$:
$R_V(u_0, u_1, u_2) = (u_0 + u_1\, x_A + u_2\, y_A)(u_0 + u_1\, x_B + u_2\, y_B)$.
It vanishes on lines passing through either A or B.

$Z(V)$ is called the *Chow variety* of V and has the advantage of being an hypersurface in the Grassmannian space, so it is determined by a unique implicit equation up to a constant factor: the Chow form R_V. Despite being determined by a unique equation, $Z(V)$ describes the variety V of unconstrained (co)dimension, see Proposition 9. Note that when V is a variety of codimension 1, we have $Z(V) \simeq V$; this explains why the theory of Chow form is effective only for codimension $c > 1$. On the other hand, the Chow form of a zero-dimensional variety $V = \{v_1, \ldots, v_k\}$ is also known as the u-resultant.

Definition 10. *Let $V \subset \mathbb{P}^n$ be a d-dimensional irreducible variety and $H_0, \ldots, H_d$ be linear forms where*

$$H_i = u_{i0}x_0 + \cdots + u_{in}x_n, \quad i = 0, \ldots, d \tag{4.2}$$

and u_{ij} are new variables, $0 \leqslant i \leqslant d$, $0 \leqslant j \leqslant n$. The Chow form R_V of V is a polynomial in the variables u_{ij} such that

$$R_V(u_{ij}) = 0 \Leftrightarrow V \cap \{H_0 = 0, \ldots, H_d = 0\} \neq \emptyset.$$

The intersection of the $d + 1$ hyperplanes H_i defined in equation (4.2) is generically a $(n - d - 1)$-dimensional linear subspace L of $\mathbb{P}^n$, i.e., an element of the Grassmannian $Gr(n - d, n + 1) = Gr(c, n + 1)$, where c is the codimension of V.

Proposition 9. [31, Prop.2.5,p.102] *A d-dimensional irreducible variety $V \subset \mathbb{P}^n$ is uniquely determined by its Chow form. More precisely, a point $\xi \in \mathbb{P}^n$ lies in V if and only if any $(n - d - 1)$-dimensional plane containing ξ belongs to the Chow variety $Z(V)$ defined by R_V.*

4.1.2 Computing and using R_V

One standard way to compute the Chow form would be to proceed as follows. Consider a variety V as in Proposition 9, parameterized as

$$x_j = f_j(t),\ j = 0, \ldots, n,\ t = (t_0 : \cdots : t_d),$$

where f_j are homogeneous polynomials of the same degree, and $d+1$ hyperplanes $H_0 = \cdots = H_d = 0$, where H_i is defined as in equation (4.2). Substituting $x_j = f_j(t)$ in every equation $H_i = 0$, we would obtain an overdetermined system of equations in the parameters t. We would then saturate these equations by the parameterization polynomials $f_j(t)$, $j = 0, \ldots, n$, that is, removing the common solutions of the parameterization (base points) e.g. by using a Gröbner basis method. This step can be ignored if V has no base point (e.g. V is a curve and $\gcd(f_0, \ldots, f_n) = 1$). Then we could eliminate the parameters t by using resultants. This reduces the computation of any Chow form to the computation of a resultant:

Corollary 10. *Consider any $V \subset \mathbb{P}^n$ of dimension d, with parameterization $x_j = f_j(t)$, $j = 0, \ldots, n$. Then, the Chow form R_V is the resultant of the hyperplane equations H_i ($0 \leqslant i \leqslant d$), where one eliminates t.*

Here the resultant should be understood, in the sense of Definition 5, as a polynomial that eliminates the variables of the input equations. In other words, while the variables of R_V are u_{ij}, its coefficients are polynomials in the coefficients of the parametric homogeneous functions f_j.

The coordinates used to represent points in the Grassmannian, and, hence, to describe R_V, are most commonly defined as the maximal minors of the $(d+1)\times(n+1)$ matrix whose rows are the normals to the hyperplanes H_i. These are known as *Plücker coordinates* or *brackets* and are the variables of R_V. Brackets are denoted as $[j_0, j_1, \ldots, j_d]$, where indices correspond to columns of the matrix. Equivalently, the *dual Plücker coordinates* or *dual brackets* can be used; these are the maximal minors of a $(n-d)\times(n+1)$ matrix whose rows are $n-d$ points that span the intersection L of the hyperplanes H_i. Dual brackets are denoted as $[[j_0, j_1, \ldots, j_{n-d-1}]]$, where indices correspond to columns of the matrix. Brackets and dual brackets with complementary index sets are equal up to sign. There are algorithms to recover the implicit or affine equations of hypersurfaces intersecting on V from R_V [31], [66].

Assuming that R_V is a polynomial in the Plücker coordinates, to obtain a representation for V as intersection of implicit hypersurfaces from its Chow form, one may apply a rewriting method. There are two such methods, namely [31, Cor.2.6,p.102], and [17, Prop.3.1], see also [66]. They are not straightforward procedures and, in the case of implicitization, typically yield more implicit polynomials than necessary. All implicit polynomials in X have the same degree as the degree of V.

To illustrate the approach in [17], let us focus on varieties of codimension 2 in $\mathbb{P}^3$, i.e., space curves. Consider the planes H_0, H_1 in equation (4.1). The Chow form R_V is a polynomial

in the brackets $[j_0, j_1]$, where $[j_0, j_1]$ denotes the maximal minor indexed by the columns $0 \leqslant j_0, j_1 \leqslant 3$, of the matrix

$$U := \begin{pmatrix} u_{00} & u_{01} & u_{02} & u_{03} \\ u_{10} & u_{11} & u_{12} & u_{13} \end{pmatrix}.$$

R_V is then rewritten as a polynomial in the dual brackets $[[j_0, j_1]]$, using the relations: $[0,1] = [[2,3]]$, $[0,2] = -[[1,3]]$, $[0,3] = [[1,2]]$, $[1,2] = [[0,3]]$, $[1,3] = -[[0,2]]$, $[2,3] = [[0,1]]$. The dual brackets are then substituted by the determinant $u_{0j_0}u_{1j_1} - u_{0j_1}u_{1j_0}$ of the corresponding minor of U. Finally, the result is expanded as a polynomial whose variables are polynomials in the $u_{10}, u_{11}, u_{12}, u_{13}$ and its coefficients are polynomials in the $u_{00}, u_{01}, u_{02}, u_{03}$. The latter polynomials are all of degree equal to the degree of V and form a system of implicit equations of V.

Example 6. *As an example, the Chow form of the twisted cubic curve with parameterization*

$$(x_0 : x_1 : x_2 : x_3) = (s^3 : s^2t : st^2 : t^3), \quad (s,t) \in \mathbb{P}^1, \tag{4.3}$$

is given by the following determinant in the primal brackets:

$$\det \begin{pmatrix} [0,1] & [0,2] & [0,3] \\ [0,2] & [0,3]+[1,2] & [1,3] \\ [0,3] & [1,3] & [2,3] \end{pmatrix},$$

which is also known as the Bézout resultant of the system of equations (4.1) *where we have substituted the X variables with the parameterization in* (4.3)*. Rewriting this determinant in the dual brackets we obtain:*

$$\det \begin{pmatrix} [[2,3]] & -[[1,3]] & [[1,2]] \\ -[[1,3]] & [[1,2]]+[[0,3]] & -[[0,2]] \\ [[1,2]] & -[[0,2]] & [[0,1]] \end{pmatrix}$$

Substituting the dual brackets by the determinant of the corresponding minors of U and collecting the terms in the variables $\{u_{0j_0}\}$ and $\{u_{1j_1}\}$, we obtain the Chow form of the twisted cubic:

$$\begin{gathered}
(u_{02}^2u_{03} - u_{01}u_{03}^2)u_{10}^2u_{12} + (u_{01}u_{02}u_{03} - u_{02}^3)u_{10}^2u_{13} + (u_{01}u_{03}^2 - u_{02}^2u_{03})u_{10}u_{11}^2 \\
+(u_{00}u_{03}^2 - u_{01}u_{02}u_{03})u_{10}u_{11}u_{12} + (3\,u_{01}u_{02}^2 - 2\,u_{01}^2u_{03} - u_{00}u_{02}u_{03})u_{10}u_{11}u_{13} \\
+(u_{00}u_{01}u_{03} - 2\,u_{00}u_{02}^2 - 3\,u_{01}^2u_{02})u_{10}u_{12}u_{13} + (u_{03}^3 - u_{00}u_{01}u_{02})u_{10}u_{13}^2 \\
+(u_{02}^3 - u_{00}u_{03}^2)u_{11}^3 + (3\,u_{00}u_{02}u_{03} - 3\,u_{01}u_{02}^2)u_{11}^2u_{12} + (u_{00}^2u_{03} - u_{01}^3)u_{12}^3 \\
+(2\,u_{00}u_{01}u_{03} - 2\,u_{00}u_{02}^2)u_{11}^2u_{13} + (3\,u_{01}^2u_{02} - 3\,u_{00}u_{01}u_{03})u_{11}u_{12}^2 \\
+(u_{00}u_{01}u_{02} - u_{00}^2u_{03})u_{11}u_{12}u_{13} + (u_{00}^2u_{02} - u_{00}u_{01}^2)u_{11}u_{13}^2 \\
+(2\,u_{01}^2u_{03} - 2\,u_{00}u_{02}u_{03})u_{10}u_{12}^2 + (u_{00}u_{01}^2 - u_{00}^2u_{02})u_{12}^2u_{13},
\end{gathered}$$

where the polynomial coefficients in the $\{u_{0j}\}$ variables are the defining implicit equations of the twisted cubic.

In the following, we adopt a practical method that avoids conversion assuming we have a parametric representation of V. We compute a polynomial whose vanishing is a necessary but not always sufficient condition for a point to lie on the variety. The algorithm we propose in Sections 4.2 and 4.3 avoids the computation of the Chow form polynomial and the need for rewriting techniques. For space curves, we shall achieve the result of Proposition 9 by computing only a few selected subsets of $Z(V)$.

4.2 Space curves

This section derives implicit representations of parametric space curves by Chow forms. Our methods avoid complex computations, such as the rewriting algorithm. In particular, we avoid the explicit computation of the Chow form and instead focus on a proper subset of the Chow variety that is enough to describe the space curve. Indeed, the Chow form of a space curve vanishes on a space line L if and only if L intersects the space curve. The method presented here provides sets of such lines, not all of them, but enough to be able to retrieve the space curve from them. This is how we proceed:

Suppose that we have a space curve V parameterized as

$$x_j = f_j(t), \quad j = 0, \ldots, 3, \quad t = (t_0 : t_1).$$

Let the line L be defined by a symbolic point $\xi = (\xi_0 : \cdots : \xi_3)$ and a sufficiently generic point $G \notin V$. Define two planes Aff(G, ξ, P_0) and Aff(G, ξ, P_1) that intersect along L, by choosing two random points P_0 and P_1 and let $H_0(x_0 : \cdots : x_3)$ and $H_1(x_0 : \cdots : x_3)$ be their respective implicit equations, as in (4.1). The coefficients of H_0 and H_1 are now *linear polynomials* in ξ. The (homogeneous) Sylvester resultant (see[14, Chapter 3, Prop.1.7]) of this system, where we set $x_j = f_j(t)$, eliminates t and returns a polynomial in ξ which vanishes on V (but not only on V), thus offering a necessary but not sufficient condition, see Algorithm A.7.

Lemma 11. *Let* $\delta = \deg f_j(t)$, $j = 0, \ldots, 3$ *and* R_G *be the Sylvester resultant of*

$$H_0(f_0(t) : \ldots : f_3(t)), \quad H_1(f_0(t) : \ldots : f_3(t)), \tag{4.4}$$

where H_0, H_1 *are defined as above. Then* R_G *is of degree* 2δ *and factors into a degree* δ *polynomial defining a surface* $\mathcal{S}_V^G \supset V$*, and a polynomial* E_L^δ*, where* E_L *is a linear polynomial defining the plane passing through points* G, P_0, P_1*.*

Proof. The degree of the Sylvester resultant in the coefficients of each of the H_0, H_1, is δ. ξ is involved linearly in the coefficients of both H_0 and H_1, since it is taken to lie in the intersection of the two planes. Hence the degree of the sought polynomial in ξ is 2δ.

It vanishes only in two cases: if ξ belongs to the plane defined by G, P_0 and P_1, or if ξ belongs to a line passing by G and intersecting V. Hence we can divide (possibly several times) the sought polynomial in ξ by the equation E_L of the plane defined by G, P_0 and P_1, thus obtaining an equation of the conical surface $\mathcal{S}_V^G$ of vertex G and directrix V. Since such a conical surface is of degree δ, its equation is R_G / E_L^δ. □

Theorem 12. *Let $f : \mathbb{P}^1 \to \mathbb{P}^3$, be a homogeneous parameterization of a space curve V and $\mathcal{S}_V^{G_k}$, $k = 1, 2, 3$ be three conical surfaces obtained by the method above with 3 different random points $G_k \notin V$. We distinguish two cases.*

1. *If V is not planar and the points G_k are not collinear, then V is the only 1-dimensional component of $\mathcal{S}_V^{G_1} \cap \mathcal{S}_V^{G_2} \cap \mathcal{S}_V^{G_3}$.*
2. *If V is contained in a plane $\mathcal{P}$ and if G_1 is not in $\mathcal{P}$, then $V = \mathcal{P} \cap \mathcal{S}_V^{G_1}$.*

Proof. Case (1). Since $\mathcal{S}_V^{G_k}$ are three different cones - or cylinders when the vertices G_k are at infinity -, they have no 2-dimensional component in common. We first reduce the problem to the case where the vertices are $P_x \coloneqq (0 : 1 : 0 : 0)$, $P_y \coloneqq (0 : 0 : 1 : 0)$ and $P_z \coloneqq (0 : 0 : 0 : 1)$. We shall prove that an algebraic space curve is the intersection of the 3 cylinders spanned by the curve itself and of directions $\vec{x}_1, \vec{x}_2$ and $\vec{x}_3$ respectively.

Let $\mathcal{C}$ be an irreducible component of $\left(\mathcal{S}_V^{G_1} \cap \mathcal{S}_V^{G_2} \cap \mathcal{S}_V^{G_3}\right)$. Since G_1, G_2, G_3 are not collinear, there exists a map $\phi \in \mathrm{PGL}(4, \mathbb{C})$ that sends G_1 to P_x, G_2 to P_y and G_3 to P_z. By linearity of ϕ (in particular, ϕ preserves alignment), we have $\phi\left(\mathcal{S}_V^{G_1}\right) = \mathcal{S}_{\phi(V)}^{P_x}$ and similar equalities for $\phi\left(\mathcal{S}_V^{G_2}\right)$ and $\phi\left(\mathcal{S}_V^{G_3}\right)$. Also, $\phi(V)$ is a homogeneous variety parameterized by $\phi \circ f$.

Thus, if $\mathcal{C} \subset \mathcal{S}_V^{G_3}$, then $\phi(\mathcal{C}) \subset \mathcal{S}_{\phi(V)}^{P_z}$. By this argument, we only have to prove that there is no homogeneous space curve $\phi(V)$ for which $\mathcal{S}_{\phi(V)}^{P_x} \cap \mathcal{S}_{\phi(V)}^{P_y} \cap \mathcal{S}_{\phi(V)}^{P_z}$ contains a different curve than $\phi(V)$. For convenience, $\phi(V)$ is denoted by V and $\phi(\mathcal{C})$ is denoted by $\mathcal{C}$ in what follows.

Remark 2. *Since V does not lie in the plane at infinity ($x_0 = 0$), we can switch to the affine setting.*

The parameterization in this affine setting is

$$g : t_1 \in \mathbb{C} \mapsto \left(\frac{f_1(1 : t_1)}{f_0(1 : t_1)}, \frac{f_2(1 : t_1)}{f_0(1 : t_1)}, \frac{f_3(1 : t_1)}{f_0(1 : t_1)}\right) \in \mathbb{C}^3.$$

For convenience, we use the same notations for both the homogeneous parameterization and its restriction to this affine setting $t_0 = x_0 = 1$.

We now have simpler expressions for our surfaces:

- $\mathcal{S}_V^{P_x} = \{(x_1, g_2(t_1), g_3(t_1)) \mid x_1, t_1 \in \mathbb{C}\}$,
- $\mathcal{S}_V^{P_y} = \{(g_1(t_1), x_2, g_3(t_1)) \mid x_2, t_1 \in \mathbb{C}\}$,
- $\mathcal{S}_V^{P_z} = \{(g_1(t_1), g_2(t_1), x_3) \mid x_3, t_1 \in \mathbb{C}\}$.

Since $\mathcal{C} \subset \mathcal{S}_V^{P_x}$, there is a (not necessarily rational) parameterization of $\mathcal{C}$ given by $q : t_1 \in \mathbb{C} \mapsto (q_1(t_1), g_2(\varphi(t_1)), g_3(\varphi(t_1)))$, where q_1 and φ are continuous piecewise smooth maps.

Remark 3. *We see that φ (resp. q_1) is not locally constant: otherwise, a part of $\mathcal{C}$ would be included in a straight line (resp. a plane), which contradicts the fact that V is not planar.*

We can thus pick a small open disc $I \subset \mathbb{C}$ such that $q_{|I}$ is injective and so, without loss of generality, we assume that $\varphi_{|I} = \mathrm{Id}_{|I}$. Also, since V is not planar and g is rational, the sets of singular points of the three maps $\pi_{x_1} : t_1 \mapsto (g_2(t_1), g_3(t_1))$, $\pi_{x_2} : t_1 \mapsto (g_1(t_1), g_3(t_1))$ and $\pi_{x_3} : t_1 \mapsto (g_1(t_1), g_2(t_1))$ are finite. Shrinking I if necessary, we assume there is no such singular point in $q(I)$.

Now, we have a local curve $q(I) = \{(q_1(t_1), g_2(t_1), g_3(t_1)) \mid t_1 \in I\}$ included in $\left(\mathcal{S}_V^{P_x} \cap \mathcal{S}_V^{P_y} \cap \mathcal{S}_V^{P_z}\right)$. Using the fact that it lies on $\mathcal{S}_V^{P_y}$, we have another parameterization of $q(I)$ given by $r = (r_1, g_2, r_3)$ with r_i injective on I and $r_i(I) \subset g_i(\mathbb{C})$. Similarly, $q(I) \subset \mathcal{S}_V^{P_z}$ gives a third parameterization $s = (s_1, s_2, g_3)$ with s_i injective on I and $s_i(I) \subset g_i(\mathbb{C})$.

Comparing s with q and r, we have $s = (s_1, g_2, g_3)$. Lastly, since π_{x_1} is regular on I, there is only one branch in $\pi_{x_1}(s(I))$ and so $s_1^{-1}(s(I)) = g_1^{-1}(s(I))$. The curve $\mathcal{C}$ is thus locally contained in V; it follows that $\mathcal{C} = V$.

Case (2). Since $G_1 \notin \mathcal{P}$, the conical surface $\mathcal{S}_V^{G_1}$ consists only of the union of lines transversal to $\mathcal{P}$: $\mathcal{S}_V^{G_1} = \cup_{x \in V} \mathrm{Line}(G_1, x)$. Each of these lines intersects $\mathcal{P}$ only in one point $x \in V$ so the curve V is exactly $\mathcal{P} \cap \mathcal{S}_V^{G_1}$. □

Note that Theorem 12 is also valid over the reals, the key argument being the existence of local smooth maps around the (dense) set of regular points.

4.3 Varieties of arbitrary codimension

In this section we generalise the construction of Section 4.2 to varieties of arbitrary codimension. Let $V \subset \mathbb{P}^n$ be a d-dimensional variety parameterized as

$$x_j = f_j(t),\ j = 0, \ldots, n,\ t = (t_0 : \cdots : t_d),$$

and $\mathcal{G} = \{G_1, \ldots, G_{n-d-1}\}$ be a set of $n-d-1$ sufficiently generic points not in V. Choose $d+1$ sets of d random points $P_i = \{P_{i1}, \ldots, P_{id}\}$, none of these points lying in V, such that for $i = 0, \ldots, d$ the points in $\mathcal{G}$ and in P_i form an affinely independent set. Let H_i, $i = 0, \ldots, d$ be the hyperplane defined as the span of the points $\xi, \mathcal{G}, P_i$. Substitute $x_j = f_j(t)$ in each H_i to obtain the system of equations

$$H_0(f_0(t) : \cdots : f_n(t)) = \ \cdots \ = H_d(f_0(t) : \cdots : f_n(t)) = 0. \tag{4.5}$$

The resultant of the polynomial system (4.5) eliminates t and returns a polynomial $R_{\mathcal{G}}$ in ξ which vanishes on V (but not only on V), thus offering a necessary but not sufficient condition, see Algorithm A.7.

There are three issues we have to examine when generalising the algorithm. We do so in the following three subsections.

4.3.1 Computing the resultant in several variables

The resultant computation is the bottleneck of the algorithm. To compute the resultant for $d = 1$, we can use Sylvester determinantal formula. For arbitrary d there exist rational formulae yielding the resultant as the ratio of two determinants, namely the Macaulay determinant [14] or the sparse resultant matrix and one of its minors [18]. These formulae are optimal for generic coefficients. For arbitrary coefficients, an infinitesimal perturbation may be applied.

Another option is to use interpolation in conjunction with information on the resultant support. This might be obtained from degree bounds on $R_{\mathcal{G}}$, as explained below, or by the computation of the monomials of the polynomials in (4.5) using software `ResPol` from [25]. The latter is analogous to the basic approach for defining the interpolation matrix by the support obtained from the resultant polytope, see Section 2.2.2.

If we choose to interpolate the resultant, an issue arises at sampling: all generated points ξ lie on V, whereas we are trying to compute a hypersurface containing V. But the kernel of M should have large dimension and, amongst the kernel vectors, we may choose one or more "small" independent vectors to define the implicit equations. We use independent vectors so as to obtain distinct surfaces. Indeed, with independent vectors, the tangent spaces of the surfaces differ and thus the surfaces intersect transversally. Here "small" may refer to the number of non-zero vector entries, or to the total degree of the monomials corresponding to its non-zero entries.

Independently of the resultant algorithm used, though, there is always an extraneous factor in the resultant that is similar to the one pinpointed by Lemma 11. Which leads to:

4.3.2 Identifying the extraneous factor in the resultant

The resultant is indeed always reducible and only one of its irreducible components is relevant for describing V. To address this issue, Lemma 11 can be generalised as follows.

Lemma 13. *Let $\delta = \deg f_j(t)$, $j = 0, \ldots, n$ and $R_{\mathcal{G}}$ be the resultant of the equations* (4.5)*. Then $R_{\mathcal{G}}$ factors into a polynomial defining a hypersurface $\mathcal{S}_V^{\mathcal{G}}$ which is of degree at most δ^d (equality holds when there are no base points) and contains V, and an extraneous factor E^p, where E is a polynomial of degree d and $p \leqslant \delta^d$.*

Proof. We define the *generalised conical hypersurface* of directrix V and vertices $\mathcal{G} = (G_1, \cdots, G_{n-d-1})$ as following:

$$\mathcal{S}_V^{\mathcal{G}} := \cup_{x \in V} \mathsf{Aff}(G_1, \ldots, G_{n-d-1}, x)$$

By abuse of notation, we shall also denote by $\mathcal{S}_V^{\mathcal{G}}$ the square-free polynomial defining the hypersurface $\mathcal{S}_V^{\mathcal{G}}$ (unique up to a constant factor).

Let us first note that $R_{\mathcal{G}}$ has a total degree in ξ of $(d+1)\delta^d$. Indeed, the degree in t of every $H_i(f_0(t) : \cdots : f_n(t))$ is δ, and the coefficients of the H_i's are linear polynomials in ξ.

The resultant of these polynomials has degree in the coefficients of each H_i bounded by δ^d, therefore total degree $\leqslant (d+1)\delta^d$, see e.g. [14, Thm.3.1].

Now, $R_{\mathcal{G}}$ vanishes if and only if the equations (4.5) have a common solution, which happens when:

i. either the hyperplanes defined $H_i = 0$, $i = 0, \ldots, d$, intersect along a linear subspace L of dimension $n-d$ and $L \cap V \neq \emptyset$, or

ii. these hyperplanes intersect along a linear subspace of dimension $> n-d$.

The first case is dealt with by the condition $\xi \in \mathcal{S}_V^{\mathcal{G}}$. It remains to prove that the second case is equivalent to $\xi \in E$ with E being a hypersurface of degree d.

In what follows, we use the theory of *exterior algebra* ([63, Chapter 10]) to compute the equations of E. Let $(\Lambda\mathcal{G}) \coloneqq G_1 \wedge \ldots \wedge G_{n-d-1}$ for convenience.

Then,

$$H_i = \xi \wedge (\Lambda\mathcal{G}) \wedge P_{i1} \wedge \ldots \wedge P_{id}, \quad \text{for } i = 0, \ldots, d.$$

Thus,

$$\bigcap_{i=0}^{d} H_i = (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{01} \wedge \ldots \wedge P_{0d}) \cdots (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{d1} \wedge \ldots \wedge P_{dd}) =$$

$$\left[\sum_{i=1}^{d} (-1)^i \det(\xi, \mathcal{G}, P_{11}, \ldots, P_{1d}, P_{0i})(\xi \wedge (\Lambda\mathcal{G}) \wedge P_{01} \wedge \ldots \wedge P_{0(i-1)} \wedge P_{0(i+1)} \wedge \ldots \wedge P_{0d})\right] \cdot$$

$$\cdot (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{21} \wedge \ldots \wedge P_{2d}) \cdots (\xi \wedge (\Lambda\mathcal{G}) \wedge P_{d1} \wedge \ldots \wedge P_{dd}).$$

By continuing developing d times the exterior product, we obtain the relation

$$\cap_{i=0}^{d} H_i = E(\xi)(\xi \wedge (\Lambda\mathcal{G})),$$

where E is a polynomial of degree d in ξ. So the resultant vanishes if $\dim(\cap_i H_i) > n-d$, that is if $E(\xi) = 0$.

Since $R_{\mathcal{G}}(\xi) = 0$ if and only if $\mathcal{S}_V^{\mathcal{G}}(\xi) = 0$ or $E(\xi) = 0$, the factor E is present and raised to a power $p \leqslant \delta^d$, in the expression of $R_{\mathcal{G}}$. □

4.3.3 How many hypersurfaces are sufficient

Computing the resultant and factoring out the extraneous factor given by Lemma 13, yields one implicit polynomial defining a hypersurface that contains the given variety. To achieve the hypothesis of Proposition 9, we must iterate for a few distinct pointsets $\mathcal{G}$ thus obtaining implicit polynomials $\mathcal{S}_V^{\mathcal{G}} = 0$ whose intersection is V.

Unfortunately, we do not yet have an a priori bound ρ for the number of equations needed to describe the variety set-theoretically as in Theorem 12. Experimental results indicate

that for curves in $\mathbb{C}^n$, $n+1$ hypersurfaces of the type $\mathcal{S}_V^{\mathcal{G}} = 0$ are required. This bound also extends to surfaces in $\mathbb{C}^4$, where 5 such hypersurfaces are sufficient. The theoretical result in [44, Chapter V] indicates that n hypersurfaces suffice for any variety in $\mathbb{C}^n$. It is not clear how to apply this result to the specific type of hypersurfaces (conical) obtained by our method.

These equations are obtained using random pointsets $\mathcal{G}$: the hypothesis of genericity is important. Indeed, each pointset $\mathcal{G}_k$ ($k = 1, \ldots, \rho$) must obviously consist of affinely independent points not in V in order to define $\mathcal{S}_V^{\mathcal{G}_k}$ properly. It is also possible that more implicit polynomials are required for specific (bad) choices of pointsets $\mathcal{G}_k$. In particular, if there is a common affine subspace $L \subset (\cap_k \mathsf{Aff}(\mathcal{G}_k))$ and if $L \cap V \neq \emptyset$, then $L \subset \left(\cap_k \mathcal{S}_V^{\mathcal{G}_k}\right)$ and the equations of these conical hypersurfaces are not defining V set-theoretically. To avoid these degenerate cases, it may be interesting to choose the random pointsets $\mathcal{G}_k$ such that any $n+1$ points picked from those pointsets are always affinely independent.

4.3.4 Degree bounds

Before stating the implicitization algorithm, we first examine the degrees of the factors of the resultant polynomial $R_{\mathcal{G}}$.

We showed that E^p appears as a factor of $R_{\mathcal{G}}$, where p is possibly very high. For curves ($d = 1$), p indeed achieves the upper bound δ^d. However, in our tests with $1 < d < n$ and in presence of base points, the factor defining $\mathcal{S}_V^{\mathcal{G}}$ also appears to a power $q > 1$ in the expression of $R_{\mathcal{G}}$:

$$\underbrace{R_{\mathcal{G}}}_{\text{degree} \leqslant \delta^d + d\delta^d} = \underbrace{(\mathcal{S}_V^{\mathcal{G}})^q}_{\text{degree} \leqslant \delta^d \times q} \times \underbrace{E^p}_{\text{degree } d \times p}$$

Note that when V is a properly parameterized curve, the inequalities become equalities and we have $q = 1$ and $p = \delta$. The algorithm works correctly on non-properly parameterized varieties. However, a non-proper parameterization decreases the degree of $\mathcal{S}_V^{\mathcal{G}}$ by some factor (the generic number of preimages) and increases the power degree q by that same factor. In practice, the extraneous factor E seems to always appear with some power p close to its upper bound δ^d.

A tighter bound on the degree of $R_{\mathcal{G}}$ can be obtained by considering sparse resultants and *mixed volumes* [14, Chapter 7]. Let

$$MV_{-i} = MV(H_0, \ldots, H_{i-1}, H_{i+1}, \ldots, H_d),\ 0 \leqslant i \leqslant d,$$

be the mixed volume of all polynomials excluding H_i. The degree of the sparse resultant in the coefficients of H_i is known to equal MV_{-i}, therefore its total degree equals $\sum_{i=0}^{d} MV_{-i}$.

For curves in $\mathbb{P}^n$, i.e., when $d = 1$, Algorithm A.7 utilizes the Sylvester determinant for computing the resultant instead of the Macaulay or sparse resultant determinant in the general case. This fact, in conjunction with Lemma 13 for determining the extraneous factors of the resultant, make the algorithm much more efficient for $d = 1$ than in the general case of arbitrary d.

4.4 Examples

In the sequel we switch from the projective to the affine setting and set $\xi = (\xi_1, \ldots, \xi_n) \equiv (x_1, \ldots, x_n)$, for emphasizing these are the implicit variables. For curves in any ambient dimension we compute the resultant directly using the Sylvester matrix, and also by interpolation, employing degree bounds relying on mixed volume. For $d > 1$ we use the Macaulay, or the sparse resultant matrix.

The Sylvester matrix leads to polynomials of degree twice the degree of the curve for any random points $\mathcal{G}$, as expected. Interpolating the resultant leads to matrices with very large kernels: on the upside, the polynomials we obtain from the kernel vectors are of degree no greater than those of the former method. Moreover, amongst them we can find a number of polynomials of small degree and, often, smaller than the degree predicted by degree bounds: these polynomials define the variety set-theoretically.

Example 7. *Consider the twisted cubic curve $V \subset \mathbb{C}^3$ affinely parameterized as:*

$$(x_1, x_2, x_3) = (t, t^2, t^3), \quad t \in \mathbb{C}.$$

An optimal system of implicit equations for V is

$$x_1^2 - x_2 = x_2^2 - x_1 x_3 = x_1 x_2 - x_3 = 0. \tag{4.6}$$

Let L be the line passing through symbolic point $\xi = (x_1, x_2, x_3)$, and generic point $G \notin V$. We define two random planes $H_1(x_1, x_2, x_3)$, $H_2(x_1, x_2, x_3)$, intersecting at L by considering additional random points $P_1, P_2 \notin L$, respectively. The Sylvester resultant of $H_1(t, t^2, t^3) = H_2(t, t^2, t^3) = 0$ is a polynomial of degree 6 in ξ which factors into the degree 3 polynomial

$$\begin{aligned} 32x_2 - 16x_3 + 56x_1x_3 + 16x_1x_2 - 80x_2^2 - 32x_1^2 - 40x_3^2 + 42x_2^3 - 2x_3x_2x_1 \\ +56x_3x_2 - 5x_3{x_2}^2 + 5{x_3}^2x_1 - 8x_3{x_1}^2 - 48{x_2}^2x_1 + 24x_2{x_1}^2 \end{aligned}$$

and the extraneous linear factor raised to the power 3 predicted in Lemma 11.

This yields a surface containing V but not of minimal degree. Repeating the procedure 3 times, the ideal of the resulting polynomials equals the ideal defined by the polynomials in (4.6).

Alternatively, we may interpolate the Sylvester resultant above. We take as predicted support the lattice points in a 3-simplex of size 6. The 84×84 matrix constructed has a kernel of dimension 65. The corresponding 65 kernel polynomials are of degrees from 2 to 6. Amongst them, we can find the three polynomials in (4.6), up to sign.

In contrast, computing the Chow form as in [17, Section 3.3] gives 16 (homogeneous) implicit equations, all of degree 3, see Example 6.

Example 8. *Consider the space curve V in the left of the figure 1.2 affinely parameterized as:*

$$(x_1, x_2, x_3) = \left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left(\frac{1-t^2}{1+t^2} \right)^2 \right), \quad t \in \mathbb{C}.$$

It is the intersection of two cylinders:

$$x_1^2 - x_3 = x_2^2 + x_3 - 1 = 0. \tag{4.7}$$

Let line L be defined from the symbolic point $\xi = (x_1, x_2, x_3)$, and "generic" point $G \notin V$. Define two random planes H_1 and H_2 that intersect at L, by choosing random points $P_1, P_2 \notin L$, respectively. Then, the Sylvester resultant of

$$H_1\left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left(\frac{1-t^2}{1+t^2}\right)^2\right), \quad H_2\left(\frac{1-t^2}{1+t^2}, \frac{2t}{1+t^2}, \left(\frac{1-t^2}{1+t^2}\right)^2\right)$$

is a polynomial of degree 8 in ξ which factors into the following degree 4 polynomial:

$$\begin{aligned} 29 + 120x_1 - 56x_2 - 182x_3 + 168x_1x_2 - 110x_1^2 - 78x_2^2 + 90x_2^2x_3 - 12x_2x_3 + \\ 70x_1^2x_3 - 18x_1x_3 - 120x_1x_2^2 + 40x_1^2x_2 + +156x_3^2 + 56x_2^3 + 49x_2^4 + 25x_1^4 - \\ 48x_3^2x_2 - 72x_3^2x_1 + 12x_3^2x_2^2 + 28x_3x_2^3 + 12x_3^2x_1^2 - 168x_2^3x_1 + 222x_2^2x_1^2 - \\ 30x_3x_1^3 - 120x_2x_1^3 - 48x_3x_2x_1 - 102x_3x_2^2x_1 + 76x_3x_1^2x_2 \end{aligned}$$

and the expected extraneous linear factor raised to the power 4. This yields a surface containing V but not of minimal degree. Repeating the procedure 3 times, we obtain three surfaces that intersect on the curve.

Alternatively, we interpolate the Sylvester resultant above using as support the lattice points in a 3-simplex of size 8. The 165×165 matrix constructed has a kernel of dimension 133. The degrees of the corresponding 133 kernel polynomials vary from 2 to 8. Amongst them, we find the two quadratic polynomials in (4.7).

The equations above were obtained by choosing random points with integral coordinates and relatively close to the curve. When we increase the range or allow for non-integral points, in order to have better chances to avoid the non-generic points, the size of the coefficients increase. For example, using random points with integral coordinates in a box of size 100 around the curve yields equations as below, where we omit most terms:

$$39834389985357559755785075 93x^4 + \cdots - 64216978805609810549754903 04.$$

Example 9. [57, Example 5.3] *Consider the space curve parameterized as $(\frac{p_1}{q}, \frac{p_2}{q}, \frac{p_3}{q})$, where:*

$$p_1 = \tfrac{28775}{134878}\,t^4 - \tfrac{64513368541235902334413817941 2317}{44618662654079434352431998399324 00}\,t^3 - \tfrac{47828026434221466944680145374491240124}{56419462326158680749256153875975210675}\,t^2$$

$$+ \tfrac{10298677641229167982949337521716055081}{87926434794013528440399200845675653 00}\,t - \tfrac{28256478577625521695889619501560610 2373}{67703354791390416899107384651170252 8100},$$

$$p_2 = \tfrac{22552733768024491856640037074192574 87}{59000744916244372025365912550039436 00}\,t^4 - \tfrac{34811048949731884289969601722962523911 9}{33851677395695208449553692325585126405 0}\,t^3$$

$$+ \tfrac{60631257463078784542819156925667898391}{96719078273414881284439120930243218300}\,t^2 + \tfrac{25293180978708547795192839718157 27}{13830842023940351964026758319783100}\,t$$

$$- \tfrac{2160066846340}{11464617073833},$$

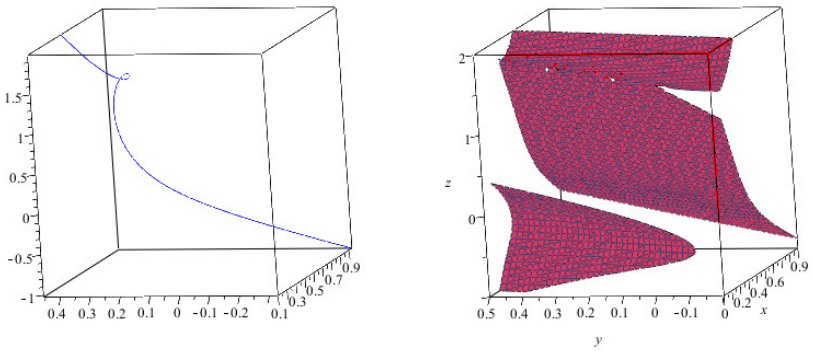


Figure 4.2: The figures depict the branch of the curve from Example 9 between the two poles (*left*) and one of the surfaces computed by the algorithm (*right*). The bottom-left part of the surface is "beyond" one of the poles and is disconnected from the other component.

$$p_3 = -\frac{1595284666744094098644242835446928142028121139 9}{1494891788945555120356185830484917011691204520 0}\,t^3 + \frac{3654698260432806341587043441984072231147356091}{1868614736181943900445232288106146264614005650}\,t^2$$

$$-\frac{1610547321878471927524238023039081161718548457}{2242337683418332680534278745727375517536806780}\,t - \frac{1248425652933171182782225268808987890080426987}{4484675366836665361068557491454751035073613560},$$

and $q = t^4 - \frac{350056078}{234205983}\,t^3 - \frac{142948855}{234205983}\,t^2 + \frac{458301406}{234205983}\,t - \frac{212187313}{234205983}$.

The denominator q has 2 real roots, which are approximately -1.143 and 1.13. Consequently, the curve has 3 connected components and so do the surfaces of the implicit equations. We computed the 3 implicit equations in 0.171 sec. Their coefficients are quite large; below we show one equation where the coefficients are rounded (see the figure 4.2).

$$1.23\,x_3 - 0.0595\,x_3^2 + 1.87\,x_3^2\,x_1\,x_2 + 78.9\,x_1^2\,x_3\,x_2 - 3.18\,x_2^2 - 26.5\,x_3\,x_1\,x_2 - 29.9\,x_3\,x_2^2\,x_1 + 2.24\,x_1 - 1.33\,x_2 - 67\,x_1^3\,x_3 + 124\,x_1^3\,x_2 - 9.1\,x_3\,x_1 + 0.266\,x_3^2\,x_1 + 4.43\,x_3\,x_2^2 + 35.7\,x_1^2\,x_3 + 48\,x_1^3 - 13.2\,x_1^2 + 3.4\,x_3\,x_2 + 13.5\,x_1\,x_2 + 1.11\,x_3^2\,x_2 + 30.4\,x_1\,x_2^2 - 67.7\,x_1^2\,x_2 - 1.32\,x_3^3\,x_2 + 0.469\,x_3^2\,x_2^2 - 0.979\,x_3^3 + 2\,x_3^3\,x_1 + 3.54\,x_3\,x_2^3 - 5.88\,x_1^2\,x_3^2 + 24.9\,x_1\,x_2^3 - 84.6\,x_1^2\,x_2^2 - 4.34\,x_2^3 + 0.354\,x_3^4 - 2.66\,x_2^4 - 65.2\,x_1^4 + 0.00316 = 0$$

Example 10. *Consider the curve in $\mathbb{C}^4$ with parameterization:*

$$(x_1, x_2, x_3, x_4) = (t^2 - t - 1,\ t^3 + 2\,t^2 - t,\ t^2 + t - 1,\ t^3 - 2\,t + 3),\ t \in \mathbb{C}.$$

The curve is defined by 5 implicit equations of degrees 1,2,2,2 and 3. Our algorithm computes 5 equations of degree 6 in 0.06 sec. These equations contain linear extraneous factors raised to the power 3. When these are divided out we obtain 5 degree 3 equations which define the curve set-theoretically.

Example 11. *We tested our method on a surface in* $\mathbb{C}^4$ *with parameterization:*

$$
\begin{aligned}
x_1 &= -1 + t_1 - 4\,t_2^2 - 5\,t_2,\\
x_2 &= 2 + 2\,t_1^2 + t_1\,t_2 - 2\,t_1 - 3\,t_2^2 - 4\,t_2,\\
x_3 &= -4 + 2\,t_1^2 - 3\,t_1\,t_2 - 2\,t_1 + 5\,t_2^2 + 3\,t_2,\\
x_4 &= 3 - 4\,t_1^2 - 4\,t_1\,t_2 - 3\,t_1 - 4\,t_2^2 + 3\,t_2.
\end{aligned}
$$

It has an implicit representation defined by 7 equations all of degree 3. When computing the resultant of the hyperplane equations in step 3 of the algorithm A.7, we used the Macaulay matrix. While the resultant of the equations is of degree 12, the Macaulay determinant is of degree 15 and factors into 4 polynomials:

$p_1 = 455\,x_3 + 750\,x_4 - 3099 + 97\,x_2 - 254\,x_1$,

$p_2 = -1231\,x_1^2 - 67428\,x_1 - 1368657 - 7911\,x_1\,x_4 - 238773\,x_4 + 120282\,x_4^2 - 7797\,x_2\,x_1 - 312183\,x_2 + 128844\,x_2\,x_4 + 30214\,x_2^2 - 13361\,x_1\,x_3 - 522219\,x_3 + 216972\,x_4\,x_3 + 98444\,x_2\,x_3 + 81846\,x_3^2$,

$p_3 = -9916630\,x_1\,x_4^3 + 9647284144\,x_2\,x_4^2 + 4038963040\,x_2^2\,x_1 + 337212316\,x_2\,x_4\,x_1^2 - 252641227648\,x_3 + 3366169952\,x_2^2\,x_3^2 + 2708928320\,x_2^3\,x_3 + 57096927668\,x_1 - 53684774940\,x_4 - 77758227688\,x_3^2 + 4536327935\,x_2\,x_3\,x_4^2 + 35030263974\,x_1\,x_3 - 122470859456\,x_2 + 3463514782\,x_2\,x_1^2 + 1675553082\,x_1\,x_2^2\,x_4 + 300052921\,x_2\,x_1^3 + 577364984\,x_1\,x_3^3 + 973095808\,x_2\,x_4^3 + 273084828\,x_4^4 + 475451312\,x_3^4 - 5750362055\,x_1^2 + 5758157904\,x_4\,x_2\,x_3^2 + 226483168\,x_1^2\,x_4^2 + 4022646056\,x_2\,x_1\,x_3\,x_4 + 17577663031\,x_4^2 + 634192304\,x_2^2\,x_3 + 1481456249\,x_2^2\,x_1^2 + 870615518\,x_1^2\,x_3^2 + 87651711\,x_1^3\,x_3 + 8195410478\,x_2\,x_3\,x_1 + 11932472704\,x_1\,x_4 + 1052921408\,x_2^4 - 20488463200\,x_2^2 + 9674045090\,x_1\,x_3\,x_4 + 5991087840\,x_4\,x_2^2\,x_3 - 5940646664\,x_2\,x_3^2 + 1965786032\,x_2\,x_3^3 - 263941933984 + 17138674\,x_1^3\,x_4 + 12449275102\,x_2\,x_4\,x_1 - 17345909250\,x_4\,x_3 + 1069083008\,x_2^3 + 707521563\,x_1^3 + 5581659148\,x_4^3 - 6156701992\,x_3^3 + 61103392\,x_1^4 + 532059077\,x_1\,x_4^2\,x_3 + 1965490737\,x_2\,x_3\,x_1^2 + 650614286\,x_1^2\,x_3\,x_4 + 2274927496\,x_1\,x_2\,x_3^2 + 3301748912\,x_1\,x_2^2\,x_3 + 1655223786\,x_1\,x_4\,x_3^2 + 982338067\,x_1\,x_2\,x_4^2 + 3614745108\,x_4^2\,x_3^2 + 1774467334\,x_4^3\,x_3 + 7406737452\,x_2\,x_1 - 17927257812\,x_2\,x_4 - 75703596848\,x_2\,x_3 + 1991048528\,x_4\,x_2^3 + 2510474960\,x_4\,x_3^3 + 10630864230\,x_2\,x_3\,x_4 + 8796789936\,x_2^2\,x_4 + 2284266257\,x_1\,x_4^2 + 7334934310\,x_1\,x_3^2 + 1319302559\,x_1^2\,x_3 + 1991564624\,x_1\,x_2^3 + 1984784289\,x_2^2\,x_4^2 + 17208929291\,x_4^2\,x_3 + 7927974534\,x_4\,x_3^2 + 2480674448\,x_1^2\,x_4$,

$p_4 = (-14670609 - 30942341\,x_3 - 11989497\,x_2 + 4731176\,x_1 - 28710187\,x_4 + 4015732\,x_3^2 + 3763632\,x_2\,x_3 + 933756\,x_2^2 - 1626083\,x_1\,x_3 - 788463\,x_2\,x_1 + 31801\,x_1^2 + 11404020\,x_4\,x_3 + 4795500\,x_2\,x_4 - 2469429\,x_1\,x_4 + 7274552\,x_4^2)^4$.

Polynomial p_3 *is irreducible of degree 4 and contains the surface;* p_4 *is a quadratic polynomial raised to the power 4; it is the extraneous factor predicted by the algorithm. The other two factors* p_1, p_2*, of degrees 1 and 2, respectively, are extraneous factors from the Macaulay's matrix construction. A total of 5 polynomials like* p_3 *define the surface set-theoretically.*

Example 12. *The algorithm A.7 works even in the presence of base points. Consider the surface in $\mathbb{C}^4$ with projective parameterization*

$$(s:t:u) \mapsto \left(\frac{p_1(s:t:u)}{q(s:t:u)}, \frac{p_2(s:t:u)}{q(s:t:u)}, \frac{p_3(s:t:u)}{q(s:t:u)}, \frac{p_4(s:t:u)}{q(s:t:u)} \right),$$

where:

$$\begin{aligned} p_1(s:t:u) &= s^2 + t^2 - u^2, \quad p_2(s:t:u) = u(s+t-u), \\ p_3(s:t:u) &= su - (t-u)^2, \quad p_4(s:t:u) = (s-u)^2 + (t-u)^2 - u^2, \\ q(s:t:u) &= st. \end{aligned}$$

Then $(0:1:1)$ and $(1:0:1)$ are two base points of the surface. The Macaulay matrix computed by the algorithm is of size 15 and of rank 13 while we expect the resultant to be of degree at most 12. Taking a non-zero minor of size 13 yields a polynomial that can be factored into the following:

- *a factor $\mathcal{S}_V^G(x_0:x_1:x_2:x_3:x_4)$ of degree 2 that contains the surface,*
- *a factor of degree 2 that is the extraneous factor predicted by the algorithm, raised to a power 3,*
- *another factor of degree 2 and two factors of degree 1, one of which is squared.*

The presence of the base points indeed reduced both the degree of the implicit equation containing the surface and the degree of the resultant. Thus we obtained more extraneous factors from the Macaulay construction.

$$\begin{aligned} &S_V^G(x_0:x_1:x_2:x_3:x_4) = x_2^2 + x_4^2 + 3x_3x_2 - 2x_4x_3 + 3x_3^2 - 2x_0x_2 + 2x_4x_0 - \\ &- 5x_0x_3 + 2x_0^2 + 3x_1x_2 - 5x_4x_1 + 9x_3x_1 - 8x_0x_1 + 8x_1^2. \end{aligned}$$

The extraneous factor predicted by the algorithm can be computed without factoring, by using the exterior algebra formulae. For the algorithm, we chose 7 random points: G and $P_{ij}, 0 \leqslant i \leqslant 2, 1 \leqslant j \leqslant 2$. The formulae is the following:

$$\begin{aligned} &(\xi \wedge G \wedge P_{01} \wedge P_{02}) \cdot (\xi \wedge G \wedge P_{11} \wedge P_{12}) \cdot (\xi \wedge G \wedge P_{21} \wedge P_{22}) = \\ &[\mathsf{det}(\xi, G, P_{11}, P_{12}, P_{01})(\xi \wedge G \wedge P_{02}) - \mathsf{det}(\xi, G, P_{11}, P_{12}, P_{02})(\xi \wedge G \wedge P_{01})] \cdot \\ &\qquad \cdot (\xi \wedge G \wedge P_{21} \wedge P_{22}) = \\ &[\mathsf{det}(\xi, G, P_{11}, P_{12}, P_{01})\, \mathsf{det}(\xi, G, P_{21}, P_{22}, P_{02}) - \\ &\qquad - \mathsf{det}(\xi, G, P_{11}, P_{12}, P_{02})\, \mathsf{det}(\xi, G, P_{21}, P_{22}, P_{01})](\xi \wedge G) \end{aligned}$$

So the extraneous factor is obtained by computing these four 5×5 determinants. Note that they can be reduced to 4×4 determinants since their last row consists of ones.

Example 13. *The algorithm A.7 can also handle parametric equations containing additional formal parameters.*

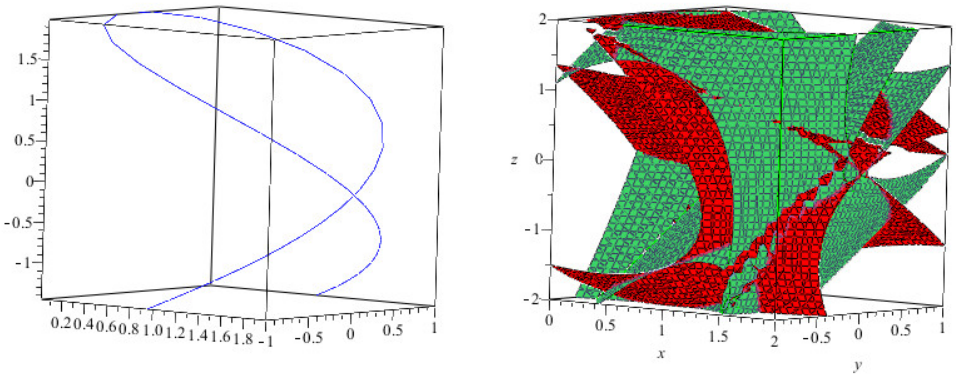


Figure 4.3: The curve of Example 13(c.) for $a = 1$ (*left*) and two of its three defining equations found by the algorithm A.7 (*right*). A third equation is needed in order to remove the extraneous surface intersections.

a. *Consider for instance a parameterized cubic:*

$$(x_1, x_2, x_3) = \left(at, bt^2, \frac{a+b}{2} t^3 \right), \; t \in \mathbb{C}.$$

The algorithm A.7 computes 3 implicit equations of degree 3 in the x_i and of degree 5 in $\{a, b\}$. Those equations match the ones obtained for the twisted cubic when $a = b = 1$.

b. *Consider the following curve of degree 4 in $\mathbb{C}^3$:*

$$\begin{aligned} x_1(t) &= -at^{12} - t^7 - bt^6 - 8t^3 - t + 9 \\ x_2(t) &= (b-3)t^{11} + 3t^{10} - at^8 - 8t^7 + 8t^6 - t \\ x_3(t) &= (a-b)t^{12} - 2t^{11} - t^9 - 3t^7 - 7t^6 + t^2 + b \end{aligned}$$

The algorithm A.7 computes 3 implicit equations of conical surfaces, of degree 12 in the x_i, in about 10h.

c. *Consider Viviani's curve parameterized as:*
$x_1 = 128\, a^5\, t^2/(256\, a^8 + 32\, a^4\, t^2 + t^4)$,
$x_2 = (256\, a^7\, t - 16\, a^3\, t^3)/(256\, a^8 + 32\, a^4\, t^2 + t^4)$,
$x_3 = (32\, a^5 - 2\, a\, t^2)\,(16\, a^4 + t^2)/(256\, a^8 + 32\, a^4\, t^2 + t^4)$.
Its implicit equations are: $x_1^2 + x_2^2 + x_3^2 = 4\, a^2$, $(x_1 - a)^2 + x_2^2 = a^2$.

The 3 implicit equations computed by the algorithm A.7 in 7,72 sec are of total degree 4 in x_i (see the figure 4.3). Since they are quite large, we show only one of them which is of degree 74 in a and omit most of its terms:

$$281474976710656\, a^{36}\,(-1048576\, a^{19}\, x_1\, x_3^2 - 4\, a^4\, x_3^2 - 393216\, a^{19}\, x_1^2\, x_3^2 + \cdots + 524288\, a^{18}\, x_3^2\, x_1^2 - 786432\, a^{18}\, x_3\, x_1^3) = 0.$$

5. SYZYGIES

5.1 The method of moving conics

The implicitization of rational plane curves, that is to say the finding on an implicit equation of a plane curve from a parameterization, has been extensively studied in the past. Besides the basic method based on a resultant computation directly from a parameterization, the method of moving lines introduced by Sederberg and Chen in [59], and developed further with the concept of μ-basis in [16], has been the more powerful and fruitful one in geometric modeling. In this section, we briefly review it with a particular emphasis on its generalisation to moving conics [60] that allows to obtain more compact matrices. Although there is no new result in this section, we believe that it sheds new light on this topic.

In what follows, we suppose that an homogeneous parameterization of a rational plane curve $\mathcal{C}$ is given over a field $\mathbb{K}$ by

$$\begin{array}{rccl} \phi: & \mathbb{P}^1 & \to \mathbb{P}^2 & \\ & (s:t) & \mapsto (f_0(s,t):f_1(s,t):f_2(s,t))\,, & \end{array} \tag{5.1}$$

where f_0, f_1 and f_2 are homogeneous polynomials in $\mathbb{K}[s,t]$ of the same degree $\delta \geq 1$. For the sake of simplicity, we assume that these polynomials have no common factor, so that the map ϕ is well defined everywhere on $\mathbb{P}^1$.

5.1.1 Moving lines

A *moving line* of degree $\nu \in \mathbb{N}$ is a polynomial of the form

$$L(s,t;x_0,x_1,x_2) = g_0(s,t)x_0 + g_1(s,t)x_1 + g_2(s,t)x_2$$

where g_0, g_1 and g_2 are homogeneous polynomials in $\mathbb{K}[s,t]$ of degree ν. For any point $(s_0 : t_0) \in \mathbb{P}^1$, $L(s_0,t_0;x_0,x_1,x_2)$ is a linear form in the variables x_0, x_1, x_2 that can be interpreted as the defining equation of a line in $\mathbb{P}^2$. This line moves when the point $(s_0 : t_0)$ varies in $\mathbb{P}^1$, hence its name. In addition, the moving line L is said to follow the parameterization ϕ if

$$L(s,t;f_0(s,t),f_1(s,t),f_2(s,t)) = g_0f_0 + g_1f_1 + g_2f_2 = 0.$$

Geometrically, this implies that the line defined in the plane by the equation $L = 0$ goes through the point $\phi(s:t) \in \mathcal{C}$.

For any integer $\nu \geq 0$, it is straightforward to compute a basis $L_1, \ldots, L_{r_\nu}$ of the vector space of moving lines of degree ν following ϕ by solving a simple linear system. We define the matrix $\mathbb{M}_\nu(\phi)$, or simply $\mathbb{M}_\nu$, as the matrix whose columns are filled with the

coefficients of the moving lines L_j with respect to the variables s, t. More precisely, $\mathbb{M}_\nu$ is defined by the matrix equality

$$(L_1\ L_2\ \cdots\ L_{r_\nu}) = (s^\nu\ s^{\nu-1}t\ \cdots\ t^\nu) \cdot \mathbb{M}_\nu. \tag{5.2}$$

It is of size $(\nu+1) \times r_\nu$ and its entries are linear forms in $\mathbb{K}[x_0, x_1, x_2]$. Therefore, it has sense to evaluate the matrix $\mathbb{M}_\nu$ at a point $p \in \mathbb{P}^2$, which we denote by $\mathbb{M}_\nu(p)$.

Proposition 14. *For all integer $\nu \geq \delta - 1$ we have $r_\nu \geq \nu + 1$ and*

$$\text{rank } \mathbb{M}_\nu(p) < \nu + 1 \iff p \in \mathcal{C}.$$

In addition, $r_{\delta-1} = \delta$ and $r_\nu > \nu + 1$ if $\nu \geq \delta$.

Proof. See [59] and [7, §2]. □

Thus, Proposition 14 shows that the matrices $\mathbb{M}_\nu$ are implicit representations of the curve $\mathcal{C}$ for all $\nu \geq \delta - 1$, in the sense that they allow to discriminate the points $p \in \mathbb{P}^2$ that belong to the curve $\mathcal{C}$. Introduced first in [59] as the method of moving lines, the matrix $\mathbb{M}_{\delta-1}$ is a particular member in the family of matrices $\mathbb{M}_\nu$, $\nu \geq \delta - 1$: it is a square matrix whose determinant gives an implicit equation of the curve $\mathcal{C}$ raised to the power the degree of ϕ [16, 7]. By the degree of ϕ we mean the number of pre-images of a general point on $\mathcal{C}$ via ϕ and over the algebraic closure $\overline{\mathbb{K}}$ of $\mathbb{K}$. In other words, this is nothing but the number of times the curve $\mathcal{C}$ is traced by the parameterization ϕ over $\overline{\mathbb{K}}$.

5.1.2 μ-basis

In the foundational paper [16], amongst other results the authors show that the matrices $\mathbb{M}_\nu$ exhibit a specific structure by introducing the concept of μ-basis.

Proposition 15. *There exists two moving lines p_1 and p_2 following ϕ such that any moving line L following ϕ can be written as*

$$L = h_1 p_1 + h_2 p_2,$$

where h_1 and h_2 are homogeneous polynomials in $\mathbb{K}[s, t]$. Such a couple of moving lines p_1, p_2 is a called a μ-basis of the parameterization ϕ.

In addition, the degrees μ_1 and μ_2 of the moving lines p_1 and p_2 only depend on ϕ and are such that $\mu_1 + \mu_2 = \delta$.

Proof. See for instance [16, 11]. □

As a consequence of this proposition, the vector space of moving lines we used to define the matrices $\mathbb{M}_\nu(\phi)$ have a simple description. More precisely, for any integer ν we have

$$\langle L_1, \ldots, L_{r_\nu} \rangle = \langle s^{\nu-\mu_1}p_1, s^{\nu-\mu_1-1}tp_1, \ldots, t^{\nu-\mu_1}p_1, s^{\nu-\mu_2}p_2, \ldots, t^{\nu-\mu_2}p_2 \rangle$$

where it is understood that the multiples of p_1, respectively p_2, disappear if $\nu < \mu_1$, respectively $\nu < \mu_2$. It follows that

$$r_\nu = \mathsf{max}(0, \nu - \mu_1 + 1) + \mathsf{max}(0, \nu - \mu_2 + 1).$$

Moreover, written in these special bases the matrices $\mathbb{M}_\nu$ exhibit a Sylvester-like block structure. In particular, in these bases the matrix $\mathbb{M}_{\delta-1}$ is nothing but the classical Sylvester matrix associated to the polynomials p_1 and p_2 with respect to the homogeneous variables s, t, denoted $\mathsf{Syl}(p_1, p_2)$. Thus, we recover the property that the resultant of these two polynomials, which is defined as the determinant of $\mathsf{Syl}(p_1, p_2)$, is equal to an implicit equation of $\mathcal{C}$ raised to the power the degree of ϕ.

Several methods have been proposed to compute a μ-basis. The first type of methods starts from a generating collection of moving lines following ϕ, namely the obvious moving lines of degree δ of the form

$$f_i(s,t)x_j - f_j(s,t)x_i, \quad 0 \leq i < j \leq 2, \tag{5.3}$$

and uses various reductions to reach iteratively a μ-basis by means of linear algebra algorithms; see e.g. [11, 36]. Another type of methods arise from the computation of normal forms of matrices over a principal ideal domain, typically the computation of a Popov form; see e.g. [51, 73]. So far, these latter methods exhibit the best theoretical complexity.

The matrix $\mathbb{M}_{\delta-1}$ is the smallest matrix that is an implicit representation of the curve in the family of matrices $\mathbb{M}_\nu$. For a general parameterization ϕ, the implicit equation of the curve is a degree δ homogeneous polynomial equation in $\mathbb{K}[x_0, x_1, x_2]$. Therefore, the matrices $\mathbb{M}_\nu$ with $\nu \leq \delta-2$ cannot yield an implicit representation of $\mathcal{C}$ because their entries are linear forms in $\mathbb{K}[x_0, x_1, x_2]$. As a consequence, to obtain more compact matrices it is necessary to introduce high-order extensions of the moving lines. Having in mind the correspondence between $\mathbb{M}_{\delta-1}$ and the Sylvester matrix $\mathsf{Syl}(p_1, p_2)$, the well-know family of (hybrid) Bézout matrices of p_1, p_2, which provides more compact matrices for the resultant, suggests to introduce quadratic forms in some entries of the matrices we consider.

5.1.3 Moving conics

As we call a moving line an equation of a line in the plane that moves as the parameter $(s : t) \in \mathbb{P}^1$ varies, we call a *moving conic* an equation of a conic in the plane whose coefficients depend on the parameter $(s : t) \in \mathbb{P}^1$. More concretely, a *moving conic* of degree $\nu \in \mathbb{N}$ is a polynomial of the form

$$Q(s,t;x_0,x_1,x_2) = g_{0,0}(s,t)x_0^2 + g_{0,1}(s,t)x_0x_1 + \\ g_{0,2}(s,t)x_0x_2 + g_{1,1}(s,t)x_1^2 + g_{1,2}(s,t)x_1x_2 + g_{2,2}(s,t)x_2^2$$

where the polynomials $g_{i,j}(s,t)$ are homogeneous polynomials of degree ν in $\mathbb{K}[s,t]$. In addition, this moving conic is said to follow the parameterization ϕ if

$$Q(s,t;f_0,f_1,f_2) = \sum_{0\leq i\leq j\leq 2} g_{i,j}(s,t)f_i(s,t)f_j(s,t) = 0.$$

Similarly to moving lines, this latter condition means geometrically that the conic defined in the plane by the polynomial Q goes through the point $\phi(s:t) \in \mathcal{C}$.

We can consider the vector space of moving conics following the parameterization ϕ of degree ν and, similarly to what we did with moving lines, build a coefficient matrix from them. However, such a matrix is useless in general because its entries are exclusively quadratic forms in $\mathbb{K}[x_0, x_1, x_2]$ and hence the determinants of its minors are always polynomials of even degree. Having in mind the (hybrid) Bézout matrix that we previously mentioned, a better option is to combine both moving lines and moving conics in a same coefficient matrix. We proceed as follows.

Pick an integer $\nu \geq 0$. As explained in §5.1.1, choosing a basis of the vector space of moving lines following ϕ of degree ν, denoted $\langle L_1, \ldots, L_{r_\nu} \rangle$, one can build the matrix $\mathbb{M}_\nu$. Now, one can consider the vector space W_ν of moving conics following ϕ of degree ν. As it turns out, each moving lines L_j gives the three moving conics $x_0 L_j$, $x_1 L_j$ and $x_2 L_j$ that all follow the parameterization ϕ. Therefore, these $3r_\nu$ moving conics obtained from the moving lines, generate a sub-vector space V_ν of W_ν. By solving a linear system and computing a nullspace, one can compute a basis of the quotient vector space W_ν / V_ν that we denote by $\langle Q_1, \ldots, Q_{c_\nu} \rangle$. Then, we define the matrix $\mathbb{MQ}_\nu(\phi)$, or simply $\mathbb{MQ}_\nu$, as the matrix satisfying to the equality

$$(L_1\ L_2\ \cdots\ L_{r_\nu}\ Q_1\ \cdots\ Q_{c_\nu}) = (s^\nu\ s^{\nu-1}t\ \cdots\ t^\nu) \cdot \mathbb{MQ}_\nu. \tag{5.4}$$

It is a matrix of size $(\nu+1) \times (r_\nu + c_\nu)$. By definition, its first r_ν columns is simply the matrix $\mathbb{M}_\nu$ whose entries are linear forms in $\mathbb{K}[x_0, x_1, x_2]$, and its last c_ν columns are built from moving conics, so its entries are quadratic forms in $\mathbb{K}[x_0, x_1, x_2]$.

We recall that μ_1 and μ_2 denote the degrees of a μ-basis of ϕ. Without loss of generality we assume that $\mu_1 \leq \mu_2$.

Proposition 16. *If $\nu \geq \mu_2 - 1$ then $r_\nu + c_\nu \geq \nu + 1$ and*

$$\text{rank } \mathbb{MQ}_\nu(p) < \nu + 1 \iff p \in \mathcal{C}.$$

In addition,

- *if $\mu_2 - 1 \leq \nu \leq \delta - 1$ then $r_\nu = 2(\nu+1) - \delta$, $c_\nu = \delta - 1 - \nu$ and the matrix $\mathbb{MQ}_\nu$ is a square matrix whose determinant is an implicit equation of $\mathcal{C}$, raised to the power the degree of ϕ,*
- *if $\nu \geq \delta - 1$ then $c_\nu = 0$ and $\mathbb{MQ}_\nu = \mathbb{M}_\nu$.*

Proof. These results will be obtained in the next section §5.1.4 by interpreting the matrices $\mathbb{MQ}_\nu$ as resultant matrices. See also [60]. □

In the case where $\mu_1 = \mu_2 = k$, hence $\delta = 2k$, the matrix $\mathbb{MQ}_{k-1}$ is a $k \times k$-matrix whose entries are all quadratic forms, and whose determinant is an implicit equation of $\mathcal{C}$, raised to the power the degree of ϕ. This is the only setting where such a fully quadratic matrix appears in the family of matrices of moving lines and conics. Notice that a general curve ϕ such that $\delta = 2k$ satisfies to $\mu_1 = \mu_2$.

5.1.4 Sylvester forms

We already mentioned that the definition of the family of matrices $\mathbb{MQ}_\nu$ is inspired by the more classical family of (hybrid) Bézout matrices of a μ-basis p_1, p_2 of ϕ. In what follows, we make explicit this comparison and exhibit in the same time a structure for the matrices $\mathbb{MQ}_\nu$. For that purpose we need to introduce the Sylvester forms.

Let p_1, p_2 be a μ-basis of the parameterization ϕ and denote by $\mu_1 \leq \mu_2$ their respective degrees. We recall that $\mu_1 + \mu_2 = \delta$. Let $\alpha := (\alpha_1, \alpha_2)$ be any couple of non-negative integers such $|\alpha| := \alpha_1 + \alpha_2 \leq \mu_1 - 1$. Since p_1 and p_2 are homogeneous polynomials in the variables s, t, one can decompose them as

$$\begin{aligned} p_1 &= s^{\alpha_1+1} h_{1,1} + t^{\alpha_2+1} h_{1,2}, \\ p_2 &= s^{\alpha_1+1} h_{2,1} + t^{\alpha_2+1} h_{2,2}, \end{aligned}$$

where $h_{i,j}(s, t; x_0, x_1, x_2)$ are homogeneous polynomials of degree $\mu_i - \alpha_j - 1$ with respect to the variables s, t and linear forms with respect to the variables x_0, x_1, x_2. Then, we define the polynomial

$$\mathsf{Syl}_\alpha(p_1, p_2) := \mathsf{Det} \begin{pmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \end{pmatrix}$$

and call it a *Sylvester form* of p_1, p_2.

Lemma 17. *For any α such that $|\alpha| \leq \mu_1 - 1$, the Sylvester form $\mathsf{Syl}_\alpha(p_1, p_2)$ is a moving conic of degree $\delta - 2 - |\alpha|$ following the parameterization ϕ. Moreover, it is independent of the choice of the polynomials $h_{i,j}$ modulo the μ-basis p_1, p_2, equivalently modulo the vector space of moving conics $V_{\delta-2-|\alpha|}$ defined in §5.1.3.*

Proof. The first assertion follows by construction and by the Cramer's rules for solving a linear system; we refer to [40, §3.10] for more details. □

It turns out that the Sylvester forms generate all the moving conics following ϕ of degree greater or equal to $\mu_2 - 1$. Taking again the notation of §5.1.3, here is the precise result.

Proposition 18. *Let ν be an integer such that $\mu_2 - 1 \leq \nu \leq \delta - 2$. Then the set of $\delta - 1 - \nu$ Sylvester forms*

$$\{\mathsf{Syl}_\alpha(p_1, p_2)\}_{|\alpha|=\delta-2-\nu} = \Big\{\mathsf{Syl}_{(\delta-2-\nu,0)}(p_1, p_2), \ldots, \mathsf{Syl}_{(0,\delta-2-\nu)}(p_1, p_2)\Big\}$$

form a basis of the quotient vector space W_ν / V_ν of moving conics of degree ν following ϕ and not generated from their corresponding moving lines, so that we have $c_\nu = \delta - 1 - \nu$. In addition, $W_{\delta-1} = V_{\delta-1}$ and hence $c_{\delta-1} = 0$.

Proof. These results follows from a duality property; we refer the reader to §2.1 and Theorem 2.9 in [8], and the references therein. See also §5.2.4. □

As a consequence of this proposition, the construction of the matrices $\mathbb{MQ}_\nu$, $\nu \geq \mu_2 - 1$, following (5.2) can be done with more specific choices of the bases of moving lines and moving conics of degree ν. As we already used in §5.1.2, the space of moving lines can be chosen such that

$$\langle L_1, \dots L_{r_\nu} \rangle = \langle s^{\nu-\mu_1} p_1, s^{\nu-\mu_1-1} t p_1, \dots, t^{\nu-\mu_1} p_1, s^{\nu-\mu_2} p_2, \dots, t^{\nu-\mu_2} p_2 \rangle.$$

Moreover, by Proposition 18 the space of moving conics can be chosen as

$$\langle Q_1, \dots Q_{c_\nu} \rangle = \langle \mathsf{Syl}_{(\delta-2-\nu,0)}(p_1,p_2), \mathsf{Syl}_{(\delta-3-\nu,1)}(p_1,p_2), \dots, \mathsf{Syl}_{(0,\delta-2-\nu)}(p_1,p_2) \rangle.$$

In this way, the matrix $\mathbb{MQ}_\nu$, $\nu \geq \mu_2 - 1$, exhibits a very particular structure: its first block of $r_\nu = 2(\nu+1) - \delta$ columns is the matrix $\mathbb{M}_\nu$, which is a Sylvester block built from the μ-basis p_1, p_2, and each of its last $c_\nu = \delta - 1 - \nu$ columns are filled with Sylvester forms of p_1 and p_2. This interpretation of the matrices $\mathbb{MQ}_\nu$, $\nu \geq \mu_2 - 1$, allows us to identify them with the family of (hybrid) Bézout matrices that are precisely defined in this way in the literature (see e.g. [20, 60]). The determinant of these Bézout matrices is known to be equal to the resultant of the μ-basis p_1, p_2. Therefore, we obtain the main property of these square matrices $\mathbb{MQ}_\nu$, $\mu_2 - 1 \leq \nu \leq \delta - 1$: their determinants are all equal to an implicit equation of the curve $\mathcal{C}$, raised to the power the degree of ϕ, as stated in Proposition 16.

In summary, the family of matrices $\mathbb{MQ}_\nu(\phi)$, $\nu \geq \mu_2 - 1$, gives implicit matrix representations of the rational curve $\mathcal{C}$. It is an extension of the family of matrices $\mathbb{M}_\nu(\phi)$, $\nu \geq \delta - 1$ with more compact matrices obtained by introducing moving conics. The more compact matrix, namely $\mathbb{MQ}_{\mu_2-1}$, is made of a Sylvester block built from the polynomial p_1, possibly empty if $\mu_1 = \mu_2$, and then filled by columns with Sylvester forms.

In the next section, we will generalise the above results to the case of rational curves in arbitrary dimension. The family of matrices $\mathbb{M}_\nu(\phi)$ built solely with moving lines, i.e. such that $\nu \geq \delta - 1$, has already been extended to this setting in [8]; we will review it briefly. The main contribution of this paper is the generalisation of the matrices built with moving conics, i.e. the matrices $\mathbb{MQ}_\nu(\phi)$ such that $\mu_2 - 1 \leq \nu \leq \delta - 2$.

5.2 The method of moving quadrics

In what follows, we suppose that an homogeneous parameterization of a rational curve $\mathcal{C} \subset \mathbb{P}^n$, $n \geq 2$, is given over a field $\mathbb{K}$ by

$$\begin{aligned} \phi : \quad \mathbb{P}^1 &\to \mathbb{P}^n \\ (s:t) &\mapsto (f_0(s,t) : f_1(s,t) : \cdots : f_n(s,t)), \end{aligned} \tag{5.5}$$

where $f_0, \dots, f_n$ are homogeneous polynomials in $\mathbb{K}[s,t]$ of the same degree $\delta \geq 1$. As in the case of plane curves, for the sake of simplicity we assume without loss of generality that these polynomials have no common factor, so that the map ϕ is well defined everywhere on $\mathbb{P}^1$.

Unlike the case of plane curves, if $n \geq 3$ a single polynomial equation in $\mathbb{K}[x_0, \ldots, x_n]$ is not sufficient to describe implicitly the curve $\mathcal{C}$. Such an equation describe an hypersurface in $\mathbb{P}^n$ and hence a collection of at least $n-1$ of them are necessary for characterizing a curve by a dimension argument, and in general more than $n-1$ equations are needed. To be more precise, consider the ring morphism

$$\begin{array}{rcl} \mathbb{K}[x_0, \ldots, x_n] & \to & \mathbb{K}[s,t] \\ x_i & \mapsto & f_i(s,t),\ i = 0, \ldots, n. \end{array}$$

The set of polynomials that are in the kernel of this map, that is to say the polynomials $P(x_0, \ldots, x_n)$ such that $P(f_0, \ldots, f_n) = 0$, is an ideal of $\mathbb{K}[x_0, \ldots, x_n]$ that is called the defining ideal of the curve $\mathcal{C}$, denoted $\mathfrak{I}_\mathcal{C}$. Choosing a finite set of generators of this ideal with a good shape and in small number is known to be a difficult task (see for instance [30, 60, 39]). In what follows, an alternative implicit representation under the form of a matrix whose entries depend on the variables $x_0, \ldots, x_n$, is presented.

5.2.1 Moving hyperplanes and μ-basis

As a straightforward generalisation of the concept of moving lines for planar curves, a *moving hyperplane* of degree $\nu \in \mathbb{N}$ is a polynomial of the form

$$H(s,t;x_0,\ldots,x_n) = g_0(s,t)x_0 + \cdots + g_n(s,t)x_n$$

where $g_0, \ldots, g_n$ are homogeneous polynomials in $\mathbb{K}[s,t]$ of degree ν. Thus, for any point $(s_0 : t_0) \in \mathbb{P}^1$, $H(s_0, t_0; x_0, \ldots, x_n)$ can be interpreted as the defining equation of a hyperplane in $\mathbb{P}^n$ that moves when the point $(s_0 : t_0)$ varies in $\mathbb{P}^1$. The moving hyperplane H is said to follow the parameterization ϕ if

$$H(s,t;f_0(s,t),\ldots,f_n(s,t)) = g_0 f_0 + \cdots + g_n f_n = 0,$$

which means geometrically that this hyperplane of equation $H = 0$ goes through the point $\phi(s:t) \in \mathcal{C}$.

For any integer ν, one can compute a basis $H_1, \ldots, H_{r_\nu}$ of the vector space (over $\mathbb{K}$) of the moving hyperplanes of degree ν following ϕ. Then, one can define a coefficient matrix $\mathbb{M}_\nu$ by means of the following equality:

$$\begin{pmatrix} s^\nu & s^{\nu-1}t & \cdots & t^\nu \end{pmatrix} \cdot \mathbb{M}_\nu = \begin{pmatrix} H_1 & \cdots & H_{r_\nu} \end{pmatrix}.$$

The matrix $\mathbb{M}_\nu$ is of size $(\nu+1) \times r_\nu$ and its entries are linear forms in $\mathbb{K}[x_0, \ldots, x_n]$, so it makes sense to evaluate it at a point in $\mathbb{P}^n$. For instance, by definition we have that for all point $(s_0 : t_0) \in \mathbb{P}^1$ this matrix satisfies to

$$\begin{pmatrix} s_0^\nu & s_0^{\nu-1}t_0 & \cdots & t_0^\nu \end{pmatrix} \cdot \mathbb{M}_\nu(\phi(s_0,t_0)) = \begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}. \tag{5.6}$$

This property implies that for any integer ν and any point $p \in \mathcal{C}$ the cokernel (or left nullspace) of $\mathbb{M}_\nu(p)$ has positive dimension. Actually, one can show that if $\nu \geq \delta - 1$ then $r_\nu > \nu + 1$ and we have that

$$\text{rank } \mathbb{M}_\nu(p) < \nu + 1 \iff p \in \mathcal{C}.$$

However, this first generalisation of Proposition 14 can be improved, but in order to state it we first need to introduce the concept of μ-basis for a parameterized curve in $\mathbb{P}^n$, $n \geq 2$, that has been introduced in [16] and then extensively studied (see e.g. [64] and [38, §4]).

Proposition 19. *There exist n moving hyperplanes $p_1, \ldots, p_n$ following ϕ such that any moving hyperplane H following ϕ can be written in the form*

$$H = h_1 p_1 + h_2 p_2 \ldots + h_n p_n,$$

where $h_1, \ldots, h_n$ are homogeneous polynomials in $\mathbb{K}[s,t]$. Such an n-tuple of moving hyperplanes $p_1, \cdots, p_n$ are called a μ-basis of the parameterization ϕ.

In addition, let $\mu_1, \ldots, \mu_n$ be the degrees of the polynomials $p_1, \ldots, p_n$ respectively and assume without loss of generality that $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$. Then, the sequence $(\mu_1, \ldots, \mu_n)$ only depends on the parameterization ϕ and $\sum_{i=1}^n \mu_i = \delta$.

Proof. See e.g. [16, §5] and [64, §2]. □

Coming back to the family of matrices $\mathbb{M}_\nu$, they have a Sylvester block structure inherited from the existence of μ-basis. In particular,

$$r_\nu = \sum_{i=1}^{n} \max(0, \nu - \mu_i + 1). \tag{5.7}$$

Moreover, we have the following generalisation of Proposition 14.

Proposition 20. *For all integer $\nu \geq \mu_n + \mu_{n-1} - 1$ we have $r_\nu > \nu + 1$ and*

$$\text{rank } \mathbb{M}_\nu(p) < \nu + 1 \iff p \in \mathcal{C}.$$

Proof. See [8]. □

As in the case of plane curves, the matrices $\mathbb{M}_\nu$ give implicit representations of the curve $\mathcal{C}$ for all ν above a certain threshold (observe that if $n = 2$ then $\mu_2 + \mu_1 - 1 = \delta - 1$). Indeed the point p on the curve $\mathcal{C}$ is characterized by the fact that the rank of such a matrix evaluated at p is not maximal. Compared to an implicit polynomial representation, this is much more efficient since only a single matrix is necessary. Moreover, these matrices allow to recover the pre-images of such points p and they are also adapted to numerical treatments by means of numerical linear algebra techniques (see [8, 6]). In what follows, we extend this family of matrices in order to obtain more compact matrices still providing an implicit representation of $\mathcal{C}$.

5.2.2 Moving quadrics

Not surprisingly, a *moving quadric* of degree $\nu \in \mathbb{N}$ is a polynomial of the form

$$Q(s,t;x_0,\ldots,x_n) = \sum_{0\le i\le j\le n} q_{ij}(s,t)x_ix_j$$

where $q_{i,j}(s,t)$, $0 \le i \le j \le n$, are $n(n+1)/2$ homogeneous polynomials in $\mathbb{K}[s,t]$. In addition, a moving quadric is said to follow the parameterization ϕ if $Q(s,t;\phi(s,t)) = 0$, hence the polynomial Q defines a quadric in space that moves with the parameter $(s:t) \in \mathbb{P}^1$ and that goes through the point $\phi(s,t) \in \mathcal{C}$.

Choose an integer ν and let $\langle H_1,\ldots,H_{r_\nu}\rangle$ be a basis of the vector space of moving hyperplanes following ϕ. We can consider the vector space W_ν of moving quadrics following ϕ. Each moving hyperplane H_j of degree ν following ϕ generates $n+1$ moving quadrics of the same degree ν, still following ϕ, that are given by x_iH_j, $0 \le i \le n$. Observe that geometrically, such a moving quadric consists of the union of the moving hyperplane of equation $H_j = 0$ and the static hyperplane of equation $x_i = 0$. We denote by V_ν the sub-vector space of moving quadrics generated by these moving quadrics obtained from moving hyperplanes. Now, let $\langle Q_1,\ldots,Q_{c_\nu}\rangle$ be basis of the quotient vector space W_ν/V_ν. Then, we define the matrix $\mathbb{MQ}_\nu(\phi)$ by

$$(H_1\ H_2\ \cdots\ H_{r_\nu}\ Q_1\ \cdots\ Q_{c_\nu}) = (s^\nu\ s^{\nu-1}t\ \cdots\ t^\nu)\cdot\mathbb{MQ}_\nu.$$

It is a matrix of size $(\nu+1)\times(r_\nu+c_\nu)$, r_ν being given by (5.7). Observe that this definition encapsulates the definition of the similar matrices we considered in the case $n=2$. By definition, the first r_ν columns of $\mathbb{MQ}_\nu$ correspond to the matrix $\mathbb{M}_\nu$ introduced in §5.2.1 and its entries are linear forms in $\mathbb{K}[x_0,\ldots,x_n]$. On the other hand, its last c_ν columns are built from moving quadrics and hence its corresponding entries are quadratic forms in $\mathbb{K}[x_0,\ldots,x_n]$. The definition of the matrices $\mathbb{MQ}_\nu$ is translated into Algorithm A.8.

We recall that the sequence of increasing integers $\mu_1 \le \mu_2 \le \ldots \le \mu_n$ denotes the degrees of a μ-basis of ϕ. Here is our first main result,

Theorem 21. *Assume that* $\nu \ge \mu_n - 1$. *Then* $r_\nu + c_\nu \ge \nu+1$ *and*

$$\text{rank } \mathbb{MQ}_\nu(p) < \nu+1 \iff p \in \mathcal{C}.$$

Moreover, we have that

$$c_\nu = \sum_{1\le i<j\le n} \max(0,\mu_i+\mu_j-1-\nu).$$

In particular, if $\nu \ge \mu_n+\mu_{n-1}-1$ *then* $c_\nu = 0$ *and it follows that* $\mathbb{MQ}_\nu = \mathbb{M}_\nu$.

The proof of this theorem is postponed to Section 5.2.4. For now, we discuss the shape of this matrix for some specific values of the degrees of the μ-basis. We emphasize that

unlike in the case of plane curves, the matrices $\mathbb{MQ}_\nu$ will never be square matrices for space curves because a space curve cannot be defined by a single equation over an algebraically closed field.

In the family of matrices $\mathbb{MQ}_\nu$, $\nu \geq \mu_n - 1$, the matrix $\mathbb{MQ}_{\mu_n - 1}$ is evidently the one with the smallest number of rows. Moreover, the smallest possible value for the integer μ_n is $\lceil \delta/n \rceil$ because of the equality $\sum_{i=1}^n \mu_i = \delta$. It corresponds to the situation where the μ_i's are evenly distributed. It turns out that this balanced situation is the generic one when $\mathbb{K}$ is an algebraic closed field: fixing a degree δ and picking n random homogeneous polynomials in (s,t) of degree δ, $f_0, \ldots, f_n$ using a dense distribution of the coefficients such as Gaussian distribution, the degrees of its μ-basis are evenly distributed with probability 1 (see [19, Theorem 1.2] for the case $n = 2$ and [16, Section 3, Theorem 1] for a proof that generalises to arbitrary dimension $n \geq 2$).

Here are some further specific settings:

- $\mu_1 = 0$: An element of degree 0 in the μ-basis corresponds to a (non-moving) hyperplane containing the curve. In this situation, we have $\mu_2 + \ldots + \mu_n = \delta$ and the problem is reduced to examining a curve in $\mathbb{P}^{n-1}$ a μ-basis of which is $(p_2, \ldots, p_n)$.
- $\mu_1 = \mu_2 = 1$: In this situation, the curve is contained in a (non-moving) quadric the equation of which is given by the resultant of p_1 and p_2.
- $\mu_i = \delta/n$ for all i: In this case, the degree δ is a multiple of n and the matrix $\mathbb{MQ}_{\delta/n-1}$ is purely quadratic since there is no moving hyperplane of degree $\delta/n - 1$ following the parameterization.

5.2.3 Computing moving quadrics using Sylvester forms

For any couple of integers $1 \leq i < j \leq n$ and any $\alpha = (\alpha_1, \alpha_2)$ such that $|\alpha| \leq \mu_i - 1$, one can consider the Sylvester form $\mathsf{Syl}_\alpha(p_i, p_j)$, as defined in §5.1.4. Similarly to Lemma 17, one can show that it is a moving quadric following ϕ of degree $\mu_i + \mu_j - 2 - |\alpha|$ that is independent of the choice of decomposition modulo the polynomials p_i, p_j.

Now, for any integer ν consider the vector space S_ν that is generated by all the Sylvester forms of degree ν, i.e.

$$\mathsf{S}_\nu = \langle \mathsf{Syl}_\alpha(p_i, p_j) \text{ such that } 1 \leq i < j \leq n \text{ and } |\alpha| = \mu_i + \mu_j - 2 - \nu \rangle .$$

Taking again the notation of §5.2.2, it is a sub-vector space of the space W_ν of moving quadrics of degree ν following ϕ. Here is our second main result.

Theorem 22. *If $\nu \geq \mu_n - 1$ then $W_\nu = V_\nu \oplus \mathsf{S}_\nu$. In other words, the moving quadrics of degree ν following ϕ are generated by the moving hyperplanes of degree ν following ϕ and by the Sylvester forms of degree ν. Moreover, these latter Sylvester forms are linearly independent and hence*

$$\dim \mathsf{S}_\nu = c_\nu = \sum_{1 \leq i < j \leq n} \max(0, \mu_i + \mu_j - \nu - 1).$$

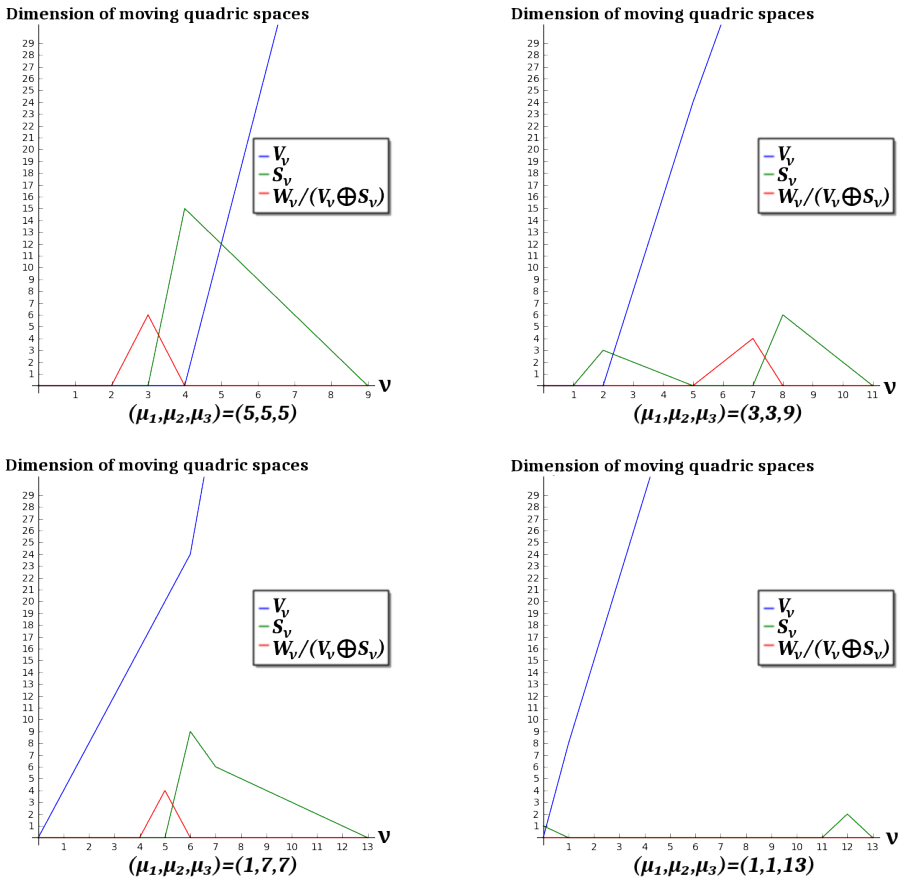


Figure 5.1: The decomposition of W_ν, for generic parametric space curves of degree 15 with fixed μ-basis degrees

This result is illustrated in Figure 5.1, where we see the dimension of $W_\nu/(V_\nu \oplus \mathsf{S}_\nu)$ vanishing at $\nu = \mu_n - 1$ in various situations.

The existence of moving quadrics following ϕ that are neither in V_ν nor in S_ν for $\nu < \mu_n - 1$ has a side effect: it may be possible for $\mathbb{MQ}_\nu$ to be an implicit matrix and satisfy the drop-of-rank property of Theorem 21 even for $\nu < \mu_n - 1$. This phenomenon appears for generic space curves of degree 7 for instance. The μ-basis of such a generic curve is balanced to degrees $\mu_1 = \mu_2 = 2$ and $\mu_3 = 3$, thus $\mathbb{MQ}_2$ is an implicit matrix of such a curve. We empirically observed that $W_1/(V_1 \oplus \mathsf{S}_1)$ is not empty and that $\mathbb{MQ}_1$ is an implicit matrix in such situation. The existence of moving quadrics in $W_\nu/(V_\nu \oplus \mathsf{S}_\nu)$ is a necessary condition for $\mathbb{MQ}_\nu$ to be an implicit matrix when $\nu < \mu_n - 1$ but is not sufficient.

The proof of this theorem is postponed to §5.2.4. Compared to Algorithm A.8 described in §5.2.2, this theorem shows that the matrices $\mathbb{MQ}_\nu$ can be computed in *closed form* in terms of the polynomials $p_1, \ldots, p_n$ defining a μ-basis of ϕ. We notice that, as far as we know, there is no known method that allows to compute the degrees $\mu_1, \ldots, \mu_n$, or even the degree μ_n, efficiently without actually computing a μ-basis. So, assuming the a μ-basis is computed, Theorem 22 gives an optimal method to build an implicit matrix representation of the curve $\mathcal{C}$ since it shows that the matrices $\mathbb{MQ}_\nu$ can be computed essentially at the cost of computing a μ-basis. This is described with more details in Algorithm A.9 for the

smallest matrix $\mathbb{MQ}_{\mu_n-1}$. Of course, a similar algorithm can be used to build the matrix $\mathbb{MQ}_\nu$ for any integer $\nu \geq \mu_n - 1$, but we prefer to focus on the smallest matrix which is the more useful in practice.

5.2.4 Proofs of the main theorems

In the case of plane curves, the proofs of Proposition 16 and Proposition 18 can be done via an identification with the classical Sylvester and hybrid Bézout matrices, relying on their well-known properties. Indeed, it is a classical result that their determinants are all equal to the resultant of a μ-basis and that this latter is equal to an implicit equation of the parameterized curve $\mathcal{C}$ (raised to the power the degree of the corresponding parameterization). In the case of space curves, the situation is more complicated and much less classical for the simple reason that a polynomial implicit representation of the curve $\mathcal{C}$ requires several polynomial equations, a set of generators of the ideal $\mathfrak{I}_\mathcal{C}$. Thus, to prove Theorem 21 and Theorem 22 we need to use some more technical tools from algebraic geometry and commutative algebra, in our view inescapable.

Moving hypersurfaces. We denote by I the ideal of the polynomial ring $R := \mathbb{K}[s,t;x_0,\ldots,x_n]$ generated by all the moving planes following ϕ. It is hence generated by a μ-basis: $I = (p_1,\ldots,p_n)$. Since we assumed that the defining polynomials $f_0,\ldots,f_n$ of the parameterization ϕ have no common root in $\mathbb{P}^1$, we deduce that the polynomials $p_1,\ldots,p_n$ have no common root in $\mathbb{P}^1$ as well [8, Lemma 1]. Algebraically, this means that they form a regular sequence [22, Chapter 17] in R outside $V(\mathfrak{m})$, the algebraic variety defined by the ideal $\mathfrak{m} := (s,t)$.

From the definitions we gave of moving hyperplanes and quadrics, it should be clear to the reader what we mean by a moving hypersurface. So, let J be the ideal of R generated by all the moving hypersurfaces, of any degree ν in (s,t) and any degree η in $x_0,\ldots,x_n$, following ϕ. Since the μ-basis is a regular sequence outside $V(\mathfrak{m})$, then J is nothing but the saturation of I with respect to $\mathfrak{m}$, that is to say:

$$J = (I :_R \mathfrak{m}^\infty) = \{p \in R : \exists k \in \mathbb{N}\ p\mathfrak{m}^k \subset I\}. \tag{5.8}$$

The ideals I and J are both bi-graded ideals. They have a grading with respect to the variables s,t and with respect to the variables $x_0,\ldots,x_n$. We denote by I_ν and J_ν the graded slices of degree $\nu \in \mathbb{N}$ with respect to the variables s,t. They are $\mathbb{K}[x_0,\ldots,x_n]$-modules [22, §0.3]. For instance, $J_0 = J \cap \mathbb{K}[x_0,\ldots,x_n] = \mathfrak{I}_\mathcal{C}$.

Elimination and matrices. We have previously built matrices by columns with the coefficients with respect to s,t of some moving hyperplanes and quadrics following ϕ of a given degree ν. Extending this approach, we could consider similar matrices built by columns with the coefficients of all the moving hypersurfaces following ϕ in a given degree ν. Call these matrices $\mathbb{MH}_\nu$. Their entries are homogeneous polynomials in $\mathbb{K}[x_0,\ldots,x_n]$. They are defined up to a choice of basis of the polynomials in s,t of degree ν, and up to a choice of a set of generators of the set of moving hypersurfaces following ϕ of degree ν.

Lemma 23. *For any integer $\nu \geq 0$ and any $p \in \mathbb{P}^n$,*

$$\text{rank } \mathbb{MIH}_\nu(p) < \nu + 1 \iff p \in \mathcal{C}.$$

Proof. Set $A := \mathbb{K}[x_0, \ldots, x_n]$. Because of (5.8), we get that the annihilator $\text{ann}_A(R_\nu/J_\nu)$ is equal to the defining ideal $\mathfrak{I}_\mathcal{C}$ of the curve $\mathcal{C}$ for all integer $\nu \geq 0$ [8, §2.3]. Then, by classical properties of Fitting ideals [22, Chapter 20], we obtain that any free presentation of R_ν/J_ν, as a A-module, has the claimed property. As J is generated by all the moving hypersurfaces following ϕ, the conclusion follows. □

Although interesting, this property is not of practical interest because it is a difficult task to compute moving hypersurfaces in general. For instance, the extreme case $\mathbb{MIH}_0$ is a row matrix filled by columns with a generating set of $\mathfrak{I}_\mathcal{C}$. Nevertheless, with this interpretation, the main idea of the method of moving hyperplanes, resp. moving quadrics, is to tune the integer ν in order to have a control on the moving hypersurfaces that are needed. Typically, one may wonder for which integer ν the moving hyperplanes, resp. quadrics, generate all the moving hypersurfaces following ϕ in this degree. Thus, Proposition 20 means that

$$\forall \nu \geq \mu_n + \mu_{n-1} - 1 \;\; J_\nu = I_\nu, \tag{5.9}$$

i.e. above this threshold degree all the moving hypersurfaces following ϕ are generated by the moving hyperplanes of the same degree following ϕ. In the same vain, to prove Theorem 21, we have to show that

$$\forall \nu \geq \mu_n - 1 \;\; J_\nu = (J\langle 2 \rangle)_\nu \tag{5.10}$$

where $J\langle 2 \rangle \subset J$ denotes the ideal of R generated by all the moving planes and moving quadrics following ϕ.

Local cohomology. A key ingredient in analyzing (5.9) and (5.10) is the local cohomology [22, Appendix 4] of the quotient ring $B := R/I$ with respect to the ideal $\mathfrak{m} = (s, t)$, denoted $H^i_\mathfrak{m}(B)$, $i \geq 0$. Indeed, by definition

$$H^0_\mathfrak{m}(B) = \{p \in B : \exists k \in \mathbb{N} \; p\mathfrak{m}^k = 0\} = (I :_B \mathfrak{m}^\infty)/I = J/I.$$

So, $H^0_\mathfrak{m}(B)$ is simply the quotient of the ideal of moving hypersurfaces following ϕ be the ideal of moving hyperplanes following ϕ. The other modules $H^i_\mathfrak{m}(B)$ are obtained as the cohomology of the Cech complex [22, Appendix 4]; it is of the form

$$\mathcal{C}^\bullet_\mathfrak{m}(B) : 0 \to B \to \oplus_{i=0}^n B_{x_i} \to \cdots \to B_{x_0 \cdots x_n}$$

where the maps are localization maps with some carefully chosen signs. They inherit from B the two gradings with respect to s, t and $x_0, \ldots, x_n$. We recall that local cohomology commutes with direct sums of modules and that the local cohomology of the polynomial ring $R = A \otimes_\mathbb{K} \mathbb{K}[s, t]$ with respect to $\mathfrak{m}$ is well known: $H^i_\mathfrak{m}(R) = 0$ if $i \neq 2$ and

$$H^2_\mathfrak{m}(R) \simeq A \otimes_\mathbb{K} \check{S}, \;\; \check{S} := \frac{1}{st}\mathbb{K}[s^{-1}, t^{-1}]. \tag{5.11}$$

For instance, we deduce that $H^2_{\mathfrak{m}}(R)_\nu = 0$ for all $\nu > -2$.

The Koszul complex. Another key ingredient to deal with the properties (5.9) and (5.10) is the Koszul complex [22, Chapter 17]. We consider the Koszul complex associated to sequence $p_1, \ldots, p_n$ that generates the ideal I. We will need to examine both gradings with respect to $\mathbb{K}[s,t]$ and to $\mathbb{K}[x_0, \ldots, x_n]$: we denote the shifts in degrees by $[-]$, resp. $\{-\}$, with respect to $\mathbb{K}[s,t]$, resp. $\mathbb{K}[x_0, \ldots, x_n]$. With this notation, this Koszul complex is of the form

$$K_\bullet : K_n \xrightarrow{d_n} \cdots \to K_2 \xrightarrow{d_2} K_1 \xrightarrow{d_1} K_0 = R$$

where $K_1 = \oplus_{i=1}^n R[-\mu_i]\{-1\}$ and $K_p = \wedge^p K_1$, the map d_1 being given by the row matrix filled with the p_i's. It is immediate to see that $H_0(K_\bullet) = R/I = B$.

Proposition 24. *With the above notation, we have an isomorphism of graded modules*

$$H_2(H^2_{\mathfrak{m}}(K_\bullet)) \xrightarrow{\sim} H^0_{\mathfrak{m}}(B) = J/I.$$

Proof. This proof uses spectral sequences [22, A.3.13]. Consider the double complex $\mathcal{C}^\bullet_{\mathfrak{m}}(K_\bullet)$ obtained from the Koszul complex $K_\bullet$ by extending each term K_p with its corresponding Cech complex $\mathcal{C}^\bullet_{\mathfrak{m}}(K_p)$. The spectral sequence corresponding to the column filtration of our double complex converges at the second step because the polynomials $p_1, \ldots, p_n$ form a regular sequence outside $V(\mathfrak{m})$. Therefore, we obtain the following terms

$$\begin{array}{ccccc} H^0_{\mathfrak{m}}(H_n(K_\bullet)) & \cdots & H^0_{\mathfrak{m}}(H_1(K_\bullet)) & H^0_{\mathfrak{m}}(H_0(K_\bullet)) \\ 0 & \cdots & 0 & H^1_{\mathfrak{m}}(H_0(K_\bullet)) \\ 0 & \cdots & 0 & 0. \end{array}$$

On the other hand, the row filtration of our double complex gives another spectral sequence that also converge at the second step; we get:

$$\begin{array}{ccccc} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ H_n(H^2_{\mathfrak{m}}(K_\bullet)) & \cdots & H_1(H^2_{\mathfrak{m}}(K_\bullet)) & H_0(H^2_{\mathfrak{m}}(K_\bullet)) \end{array}$$

From here, since $H_0(K_\bullet) = B$, the claimed isomorphism follows from the fact that these two spectral sequences converge to the same limit, namely the homology of the total complex of $\mathcal{C}^\bullet_{\mathfrak{m}}(K_\bullet)$. □

Corollary 25. *Assume that $\nu \geq \mu_n - 1$, then we have the following exact sequence of graded A-modules*

$$\bigoplus_{1 \leq i < j < k \leq n} \check{S}_{\nu - \mu_i - \mu_j - \mu_k} \otimes_{\mathbb{K}} A\{-3\} \to \bigoplus_{1 \leq i < j \leq n} \check{S}_{\nu - \mu_i - \mu_j} \otimes_{\mathbb{K}} A\{-2\} \to (J/I)_\nu \to 0.$$

Proof. The homology module $H_2(H^2_{\mathfrak{m}}(K_\bullet))$ is computed as follows. First, applying the functor $H^2_{\mathfrak{m}}(-)$ to the Koszul complex $K_\bullet$ we get the sequence

$$H^2_{\mathfrak{m}}(K_3) \xrightarrow{d_3} H^2_{\mathfrak{m}}(K_2) \xrightarrow{d_2} H^2_{\mathfrak{m}}(K_1) \tag{5.12}$$

where the maps are induced by those of the Koszul complex $K_\bullet$. Then, $H_2(H^2_{\mathfrak{m}}(K_\bullet))$ is simply Ker $d_2/\mathrm{Im}\, d_3$. Now, since $K_1 = \oplus_{i=1}^n R[-\mu_i]\{-1\}$, by (5.11) we deduce that

$$H^2_{\mathfrak{m}}(K_1)_\nu \simeq \oplus_{i=1}^n \check{S}_{\nu-\mu_i} \otimes_{\mathbb{K}} A\{-1\}.$$

In particular, we deduce that for all $\nu > \mu_n - 2$, $H^2_{\mathfrak{m}}(K_1)_\nu = 0$. Therefore, we deduce that $(\mathrm{Ker}\ d_2)_\nu = H^2_{\mathfrak{m}}(K_2)_\nu$. From here, the claimed result follows by noting that

$$K_2 = \oplus_{1\leq i<j\leq n} R[-\mu_i - \mu_j]\{-2\},$$

$$K_3 = \oplus_{1\leq i<j<k\leq n} R[-\mu_i - \mu_j - \mu_k]\{-3\},$$

and applying again (5.11). □

Theorem 21 follows straightforwardly from this corollary. Indeed, it shows that J_ν is generated by moving quadrics modulo the moving planes, i.e. modulo I_ν, and that the number of minimal generators is precisely given by c_ν. In particular, if $\nu \geq \mu_n + \mu_{n-1} - 1$ we get that $(J/I)_\nu = 0$, i.e. that $J_\nu = I_\nu$.

Duality and Sylvester forms. The proof of Theorem 22 can be seen as a particular case of an explicit construction of duality isomorphism similar to the one we obtained in Proposition 24. Such an explicit construction already appeared in [41] and [13]. It is beyond the scope of this paper to give all the details about this construction, but we mention the main steps to prove Theorem 22.

First, by Koszul self-duality [22, Proposition 17.15], we have a graded isomorphism

$$H_2(H^2_{\mathfrak{m}}(K_\bullet)) \simeq H_{n-2}(K_\bullet[d-2])^*$$

where $(-)^*$ stands for the dual. Then, one can consider the generalised Morley form that appears in [41, Section 3] and that gives an explicit construction of the map in Proposition 24, via the above isomorphism. Then, to obtain Theorem 22 one has to show that for all degree $\nu \geq \mu_n - 1$ the graded components of this Morley form coincide with Sylvester forms. This latter property follows from [40, Proposition 3.11.13] (see also [8, Lemma 2.8]).

The Koszul syzygies. Before closing this section, we discuss the link with the obvious moving hyperplanes of the form (5.3) that are also called Koszul syzygies. Let us denote by $I_{\mathcal{K}}$ the ideal generated by these moving hyperplanes. We have $I_{\mathcal{K}} \subset I \subset J$. As the polynomials $f_0, \ldots, f_n$ have no common root in $\mathbb{P}^1$, we know that these three ideals coincide in sufficiently high degrees. Here is a more precise result.

Proposition 26. *For all $\nu \geq \delta + \mu_n + \mu_{n-1} - 1$ we have $(I_{\mathcal{K}})_\nu = I_\nu$.*

Proof. The quotient $I/I_{\mathcal{K}}$ is canonically identified with the first homology group H_1^f of the Koszul complex associated to the sequence $f_0, \ldots, f_n$ which is of the form

$$K^f_{n+1} \to \cdots \to K^f_2 \xrightarrow{d_2} K^f_1 \xrightarrow{d_1} K^f_0.$$

Indeed, the kernel of d_1 corresponds to the moving planes following ϕ and the image of d_2 identifies to the obvious moving hyperplanes. Taking into account the shifts in the grading, we get the isomorphism $(H_1^f)_{\nu+\delta} \simeq (I/I_{\mathcal{K}})_\nu$ for all integer ν.

Now, consider the sequence

$$0 \to Z_2^f \hookrightarrow K_2^f \xrightarrow{d_2} K_1^f \xrightarrow{d_1} K_0^f$$

where $Z_2^f = \text{Ker } d_2$. Then, playing as in the proof of Proposition 24 with the two spectral sequences associated to the double complex

$$0 \to \mathcal{C}_{\mathfrak{m}}^\bullet(Z_2^f) \hookrightarrow \mathcal{C}_{\mathfrak{m}}^\bullet(K_2^f) \xrightarrow{d_2} \mathcal{C}_{\mathfrak{m}}^\bullet(K_1^f) \xrightarrow{d_1} \mathcal{C}_{\mathfrak{m}}^\bullet(K_0^f),$$

we deduce that $(H_1^f)_\nu = 0$ for any integer ν such that $H^2_{\mathfrak{m}}(Z_2^f)_\nu = 0$.

The two modules Z_2^f and Z_1^f are free graded $\mathbb{K}[s,t]$-modules. Consider the canonical map $\wedge^2 Z_1^f \to Z_2^f$. Since the f_i's have no common root in $\mathbb{P}^1$, we deduce that the kernel and the cokernel of this map are supported on $V(\mathfrak{m})$, and therefore it must be an isomorphism, moreover graded. To conclude, we notice that $Z_1^f \simeq \oplus_{i=1}^n \mathbb{K}[s,t](-\delta-\mu_i)$, and the claimed result follows by (5.11). □

5.2.5 Summary

To summarize, we have built a family of matrices $\mathbb{MQ}_\nu$ that provides implicit matrix representations of a parameterized curve in arbitrary dimension for all $\nu \geq \mu_n - 1$, where μ_n is the highest degree of a polynomial in a μ-basis of the parameterization of this curve. They have the following shape:

- If $\mu_n - 1 \leq \nu \leq \mu_n + \mu_{n-1} - 2$, then $\mathbb{MQ}_\nu$ is filled with moving planes and moving quadrics. It is exclusively filled with moving quadrics if and only if $\nu = \mu_n - 1$ and $\mu_i = \delta/n$ for all $i = 1, \ldots, n$.
- If $\nu \geq \mu_n + \mu_{n-1} - 1$, then $\mathbb{MQ}_\nu$ is filled with moving planes, and it coincides with the family of matrices $\mathbb{M}_\nu$ introduced in [8].
- If $\nu \geq \delta + \mu_n + \mu_{n-1} - 1$, then $\mathbb{MQ}_\nu = \mathbb{M}_\nu$ can be filled from the obvious moving planes of the form (5.3) without relying on the computation of a μ-basis. This is an improvement of [8, Proposition 26].

Example 14. *Consider the following parameterization ϕ of a curve $\mathcal{C}$ of degree 6:*

$$\begin{aligned}
f_0(s,t) &= 3s^4t^2 - 9s^3t^3 - 3s^2t^4 + 12st^5 + 6t^6,\\
f_1(s,t) &= -3s^6 + 18s^5t - 27s^4t^2 - 12s^3t^3 + 33s^2t^4 + 6st^5 - 6t^6,\\
f_2(s,t) &= s^6 - 6s^5t + 13s^4t^2 - 16s^3t^3 + 9s^2t^4 + 14st^5 - 6t^6,\\
f_3(s,t) &= -2s^4t^2 + 8s^3t^3 - 14s^2t^4 + 20st^5 - 6t^6.
\end{aligned}$$

The computation of a μ-basis of ϕ gives

$$\begin{aligned} p_1 &= (s^2 - 3st + t^2)x + t^2 y \\ p_2 &= (s^2 - st + 3t^2)y + (3s^2 - 3st - 3t^2)z, \\ p_3 &= 2t^2 z + (s^2 - 2st - 2t^2)w, \end{aligned}$$

so that we have $\mu_1 = \mu_2 = \mu_3 = 2$.

This example is taken taken from [39, Example 3.7] where the authors introduce three quartic surfaces in order to get an implicit representation of the curve $\mathcal{C}$. The equations of these quartic surfaces are given by the resultant of p_1 and p_2, of p_1 and p_3, and of p_2 and p_3 with respect to the homogeneous variables s and t. Their intersection always contains the curve $\mathcal{C}$ but it may also contains some extraneous components. For instance, in this example the point $q = (1 : 1 : 1 : 1) \in \mathbb{P}^3$ is not on the curve $\mathcal{C}$, but it belongs to the intersection of these three quartic surfaces.

In [8, Example 8], this same parameterization is implicitized by means of the matrix of moving hyperplanes $\mathbb{M}_3$ ($\mu_2 + \mu_3 - 1 = 3$), which is of size 4×6. This matrix is proved to always give an implicit representation of the curve $\mathcal{C}$. Indeed, its rank is equal to 4 after evaluation at the point q, showing that $q \notin \mathcal{C}$.

Now, according to the new family of matrices we built in this paper, the matrix of $\mathbb{MQ}_1$ ($\mu_3 - 1 = 1$) also provides an implicit representation of the curve $\mathcal{C}$. It is a matrix of size 2×6, more compact than $\mathbb{M}_3$, which is filled with the 6 Sylvester forms $\mathsf{Syl}_{(1,0)}(p_i, p_j)$ and $\mathsf{Syl}_{(0,1)}(p_i, p_j)$ for $1 \leq i < j \leq 3$. $\mathbb{MQ}_1$ is printed below.

$$\begin{pmatrix} 2xy - y^2 - 6xz - 3yz & 2xy + 6xz & 2xz - 3xw - yw \\ -8xy + y^2 + 12xz + 3yz & 2xy - y^2 - 6xz - 3yz & -6xz + 8xw + 2yw \\ xw & 2yz + 6z^2 - 5yw - 3zw & -yw - 3zw \\ 2xz - 3xw - yw & -2yz - 6z^2 + 8yw & 2yz + 6z^2 - 5yw - 3zw \end{pmatrix}$$

Matrix $\mathbb{MQ}_1$ of moving quadrics corresponding to the space curve parameterization discussed in Example 14.

5.3 Computational aspects

In this section, we report on some experiments on the computation of the family of matrices $\mathbb{MQ}_\nu$ we have introduced. In particular, we illustrate the gain we obtain with the smallest matrix $\mathbb{MQ}_{\mu_n - 1}$ for deciding if a point belongs to a parameterized curve.

We emphasize that all the applications that are discussed in [8] with the matrices of moving hyperplanes also apply with our extended family of matrices built with moving hyperplanes

and moving quadrics. For instance, the curve/curve intersection problem and the computation of the self-intersection locus of a parameterized curve can be solved with these new matrices following essentially the same algorithms; we refer the reader to [8] for more details.

5.3.1 Computation of μ-basis

To take the best advantage of the family of matrices $\mathbb{MQ}_\nu$, $\nu \geq \mu_n - 1$, it is necessary to compute a μ-basis of the input parameterized curve. Actually, we could argue that computing only the highest degree μ_n of a μ-basis would be enough for us, but as far as we know, there is no known method that allows to compute μ_n without computing an entire μ-basis. For the sake of completeness, we recall very briefly, and give references, for the three known types of methods for computing a μ-basis (see also [38]).

The first dedicated algorithm for computing a μ-basis appeared in [11, Algorithm 3.2], in the case of plane curves. Later, it has been generalised to the case of space curves in arbitrary dimension in [64, §3]. The method consists in considering the obvious moving hyperplanes (5.3) (or Koszul syzygies) and then to apply Gaussian elimination techniques in order to iteratively reduce these moving hyperplanes to a μ-basis.

Another approach for computing μ-bases comes from the methods and algorithms that are independently developed in order to compute canonical forms of univariate polynomial matrices. Thus, a μ-basis can be efficiently computed as a Popov form of a matrix built again from the obvious moving hyperplanes (5.3). As far as we know, the best complexity algorithm is described in [73]; for further details about Popov forms, we refer the reader to [70, 49].

Finally, we mention that a third approach for computing μ-bases has been recently given in [36]. It also relies on matrix reductions, but here a finer (partial) reduced row-echelon form is used.

5.3.2 Computation of the matrices

In this paragraph we report on the size and the computation time of some implicit matrix representations that are of particular interest, in the case $n = 3$. More precisely, we retain the following matrices:

- $\mathbb{M}$: a moving hyperplane matrix. It is considered either in degree $\delta - 1$, in order to avoid the computation of a μ-basis, or in its optimal degree $\mu_n + \mu_{n-1} - 1$, in which case (the degrees of) a μ-basis must be computed.
- $\mathbb{MQ}_{\mathsf{Ker}}$: the matrix of moving planes and moving quadrics in degree $\mu_n - 1$, computed using kernel calculations by Algorithm A.8.
- $\mathbb{MQ}_{\mathsf{Syl}}$: the matrix of moving planes and moving quadrics in degree $\mu_n - 1$ built in closed form from a μ-basis, by means of Algorithm A.9.

The results are reported below. The algorithms have been implemented in SageMath and run using an Intel(R) Pentium(R) N3540 CPU @ 2.16GHz on a x64 machine with 4GB of RAM.

In Table 5.1, we give the computation time of a μ-basis and then our two options to build an optimal implicit matrix representation: a matrix fully composed of moving planes or a mixed matrix with moving planes and moving quadrics. For these two matrices, the computation time excludes the computation of the μ-basis, which is reported in the second column. It appears clearly that the matrix with moving quadrics is more expensive to build, because its entries require calculations.

Degree δ and degrees $(\mu_i)_i$	**μ-basis**	$\mathbb{M}_{\mu_n+\mu_{n-1}-1}$	$\mathbb{MQ}_{\mathbf{Syl},\mu_n-1}$
$5 \quad (2,3)$	230ms	5x5 57ms	3x3 417ms
$10 \quad (5,5)$	343ms	10x10 168ms	5x5 1503ms
$10 \quad (1,9)$	292ms	10x10 166ms	9x9 614ms
$5 \quad (1,2,2)$	156ms	4x7 94ms	2x5 676ms
$9 \quad (3,3,3)$	151ms	6x9 141ms	3x9 2194ms
$9 \quad (1,4,4)$	292ms	8x15 268ms	4x9 1900ms
$9 \quad (1,1,7)$	396ms	8x15 244ms	7x14 1132ms
$15 \quad (5,5,5)$	281ms	10x15 332ms	5x15 5516ms
$15 \quad (1,7,7)$	647ms	14x27 782ms	7x15 4663ms
$15 \quad (1,1,13)$	1477ms	14x27 657ms	13x26 2810ms

Table 5.1: Computation time in milliseconds of a μ-basis and two typical implicit matrix representations built from the μ-basis.

In the Table 5.2, we assume that a μ-basis is unknown and then compare the computation time of the matrix $\mathbb{M}_{\delta-1}$, which does not require the computation of a μ-basis, with the computation time of the matrix $\mathbb{MQ}_{\mu_n-1}$ via our two algorithms, for which a μ-basis is computed. As expected, the faster matrix to compute is $\mathbb{M}_{\delta-1}$.

In summary, it appears that the new matrix $\mathbb{MQ}_{\mu_n-1}$ is not easier to build compared to the other matrices that are already known, but their computation time remains acceptable. It turns out that these implicit matrix representations are only computed once for a curve and is then stored. So in the end, the computation of the matrix itself is not the most important feature, what is the most important is the efficiency of a matrix when one computes intensively on the curve with it. In the next paragraph, we illustrate this property with the point/curve intersection problem, i.e. by testing whether a given point belongs to the curve.

Degree δ and degrees $(\mu_i)_i$	$\mathbb{M}_{\delta-1}$	$\mathbb{MQ}_{\mathbf{Ker},\mu_n-1}$	$\mathbb{MQ}_{\mathbf{Syl},\mu_n-1}$
$5 \quad (2,3)$	74ms	305ms	431ms
$10 \quad (5,5)$	226ms	409ms	1113ms
$10 \quad (1,9)$	187ms	1055ms	614ms
$5 \quad (1,2,2)$	120ms	319ms	663ms
$9 \quad (3,3,3)$	312ms	458ms	1914ms
$9 \quad (1,4,4)$	384ms	987ms	1912ms
$9 \quad (1,1,7)$	304ms	2815ms	1150ms
$15 \quad (5,5,5)$	931ms	1358ms	5989ms
$15 \quad (1,7,7)$	701ms	2311ms	4363ms
$15 \quad (1,1,13)$	946ms	8947ms	2526ms

Table 5.2: Comparison of the computation time to build the matrix $\mathbb{M}_{\delta-1}$ with the computation times of the two algorithms corresponding to build the moving quadric matrices either from kernel computation or by instantiation of Sylvester forms.

As we will see, for this use the matrices of moving quadrics we introduce behave much better than the previously known matrices.

5.3.3 The drop-of-rank property

What makes the matrices $\mathbb{MQ}_\nu$, $\nu \geq \mu_n - 1$, implicit representations is the *drop-of-rank property*: evaluated at a point p, their rank drops, more precisely their rows are linearly dependent, if and only if the point p is on the curve. This property gives a very efficient method to decide if a point belongs to a curve or not.

In Table 5.3, we compare the computation time for testing if a point belongs to a curve by means of the two moving hyperplanes matrices, $\mathbb{M}_{\delta-1}$ which is computed without μ-basis and $\mathbb{M}_{\mu_n+\mu_{n-1}-1}$ that requires the computation of a μ-basis, and by means of the smallest matrix of moving hyperplanes and quadrics we obtained, namely $\mathbb{MQ}_{\mu_n-1}$. In all cases we tested, whatever the repartition of the degrees μ_i of the μ-basis, this matrix $\mathbb{MQ}_{\mu_n-1}$ was always more efficient.

Degree δ and degrees $(\mu_i)_i$	$\mathbb{M}_{\delta-1}$	$\mathbb{M}_{\mu_n+\mu_{n-1}-1}$	$\mathbb{MQ}_{\mu_n-1}$
$5 \quad (2,3)$	54ms	54ms	22ms
$10 \quad (5,5)$	230ms	230ms	62ms
$10 \quad (1,9)$	230ms	230ms	121ms
$5 \quad (1,2,2)$	105ms	61ms	22ms
$9 \quad (3,3,3)$	353ms	125ms	59ms
$9 \quad (1,4,4)$	393ms	267ms	78ms
$9 \quad (1,1,7)$	362ms	256ms	171ms
$15 \quad (5,5,5)$	1139ms	377ms	167ms
$15 \quad (1,7,7)$	1127ms	929ms	199ms
$15 \quad (1,1,13)$	1086ms	894ms	534ms

Table 5.3: Average time over a hundred random points for testing if a point belongs to the curve.

We notice that deciding whether a point in space belongs to a parameterized curve can be done via a greatest common divisor (GCD) computation once a μ-basis is known. Indeed, let p_1, p_2, p_3 be a μ-basis of a curve parameterization, let q be a point in space and denote by $p_i(q)$ the evaluation of p_i at the point q. Then, the GCD of the three homogeneous polynomials $p_1(q)$, $p_2(q)$ and $p_3(q)$ is a homogeneous polynomial in the variables s, t whose degree is equal to the multiplicity of the point q with respect to the curve, in particular this degree is non-zero if and only if the point q belongs to the curve [71, Theorem 6.4]. However, this method requires exact computations and hence it does not allow to deal with approximate input data. In addition, the use of exact computations makes the computation time strongly dependent on the choice of the point q. To be more concrete, we applied this method to the case of the degree 9 curve with μ-basis of type $(3, 3, 3)$ that is used in Table 5.3. The points are chosen on the curve with five significant digits and are cast to rational numbers for the GCD computation. We observed an average time over a hundred random points of 66 121ms and especially a very high standard deviation of 66 593ms (with a minimum of 15ms and a maximum computation time of 176 136ms). When the matrix $\mathbb{MQ}_2$ is used we observe a standard deviation of 7ms, showing a computation time which is almost independent of the point q. This difference is mostly due to the fact that the matrices of moving hyperplanes and moving quadrics allow to rely on numerical linear algebra tools and are thus capable to deal with approximate data and computations.

To conclude, we illustrate that given a point $p \in \mathcal{C}$, not only the rank of $\mathbb{MQ}_\nu(p)$, $\nu \geq \mu_n - 1$, drops but also its cokernel (left nullspace) allows to recover all the parameters $(s_0 : t_0) \in \mathbb{P}^1$ such that $\phi(s_0, t_0) = p$, following the approach developed in [8, 6] with the matrices of moving hyperplanes.

Example 15. *Consider the lemniscate-like space curve $\mathcal{C}$ given by*

$$\begin{aligned} f_0(s,t) &= (t^2+s^2)(t^4+s^2),\\ f_1(s,t) &= t(t^2-s^2)^2,\\ f_2(s,t) &= t(t^4-s^4),\\ f_3(s,t) &= 3s^4+t^4. \end{aligned}$$

This curve has a self-intersection point at $p := (1 : 0 : 0 : 1)$. The matrix of moving quadrics $\mathbb{MQ}_2$ is of size 3×6 and, when evaluated at p, has a cokernel given by

$$v_1 = (v_{1,1}, v_{1,2}, v_{1,3}) = (1, 0, 1)$$

and

$$v_2 = (v_{2,1}, v_{2,2}, v_{2,3}) = (0, 1, 0).$$

None of these vectors are of the form $v = (s^2, st, t^2)$ but they are linear combinations of the two vectors corresponding to the evaluation of the form v at the two pre-images parameters of p. Therefore, to retrieve these two pre-images one can solve the eigenvalue problem

$$\mathsf{rank}(t\Delta_0 - s\Delta_1) < 2$$

where

$$\Delta_0 = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix}, \quad \Delta_1 = \begin{pmatrix} v_{1,2} & v_{1,3} \\ v_{2,2} & v_{2,3} \end{pmatrix}.$$

We deduce that the pre-images of p correspond to the parameters $(s_0 : t_0) = (1 : 1)$ and $(s_1 : t_1) = (1 : -1)$.

Finally, we notice that the matrix $\mathbb{MQ}_1$ is of size 2×6 and satisfies to the drop-of-rank property. Its rank drops by 2 after evaluation at p, thus it is equal to the null matrix when evaluated at p. Therefore, in this case the matrix is too small to allow the inversion of a multiple point and hence it is necessary to increase the degree ν by one. In general a matrix $\mathbb{MQ}_\nu$ allows to invert points having at most ν pre-images.

6. METHOD COMPARISONS

This section presents the features of different algorithms, either standard or presented in this work, and their practical performances through a number of examples.

In this chapter, we dismiss the case of varieties of dimension 0 and, unless specified otherwise, homogeneous parameterizations of projective varieties can be used in place of rational parameterizations of algebraic varieties.

The different algorithms discussed here are the followings.

(MPU) The algorithm A.2 of cuboid-based approximate surface interpolation presented in the section 3.1.1.

(Slim) The algorithm A.4 of sphere-based approximate surface interpolation presented in the section 3.1.2.

(Geom) A geometric primitive extraction algorithm based on RANSAC as described in [58].

(IM) The algorithm A.1 of interpolation matrix presented in the section 2.3.

(Impl) A polynomial interpolation algorithm based on the Maple command `[algcurves]implicitize` (see [12]).

(CF) The algorithm A.7 based on Chow forms presented in Chapter 4.

(GB) A Gröbner basis algorithm based on the Maple command `[PolynomialIdeals]EliminationIdeal`.

(MQ) The algorithm A.8 of moving quadrics based on syzygies and presented in the chapter 5.

The list above is far from being an exhaustive list of implicitization algorithms. It is meant both to represent the work of the author and the different approaches in this field.

As seen in Chapter 5, there are several syzygy-based matrix representations. We refer to the section 5.3 for a comparison between the different matrix representation algorithms, their advantages and their drawbacks. There are several Gröbner basis implicitization algorithms; we have chosen one that does not necessarily produces reduced Gröbner bases because it is faster. The drawback is that the number of equations given by (GB) may be not minimal.

6.0.1 Differences in the objectives

In the table 6.1, we list the type of inputs and outputs of each algorithm.

The first distinction between the implicitization algorithms considered is whether they generate a shape or compute an implicit representation of an existing shape.

	Type of input	Type of output	Dimension(s)	Exact/ Approximate
(MPU)	Point cloud	Local implicit polynomials	Surfaces in 3D*	Approximate
(Slim)	Point cloud	Local implicit procedures	Surfaces in 3D*	Approximate
(Geom)	Point cloud	List of basic shapes	Surfaces in 3D*	Approximate
(IM)	Point cloud or parameterization	Implicit matrix	Any	Exact
(Impl)	Parameterization	Implicit polynomials	Any	Exact
(CF)	Rational parameterization	Implicit polynomials	Codimension > 1	Exact
(GB)	Rational parameterization	Gröbner basis	Any	Exact
(MQ)	Rational parameterization	Implicit matrix	Curves**	Either exact or numerical

(*) Possibly generalisable to hypersurfaces
(**) Possibly generalisable to any dimension

Table 6.1: Framework of different implicitization algorithms

In the first situation, illustrated by (MPU), (Slim) and (Geom), the input is usually a point cloud and is not enough to define the target shape uniquely. The shape is thus generated in the process according to some contraints. In (MPU), the shape generated is piecewise quadratic minimizing a regularized ℓ^2 distance, the Taubin distance (see the equation (3.1)); in (Slim), the shape is defined by weighted sums of quadratic polynomials minimizing a weighted ℓ^2 distance (see the equation (3.4)); in (Geom), the shape is a union of plane, sphere, cone, cylinder and torus parts that fit parts of the point cloud within an admissible deviation. Although they produce shapes that may not be algebraic or even piecewise algebraic, these algorithms use low-degree polynomials (usually 1 or 2 even though the torus parts of (Geom) are of degree 4).

In the second situation, illustrated by (CF), (GB) and (MQ), the purpose is not to generate a shape but rather to obtain an implicit representation of an uniquely defined shape (usually given by a parameterization). These algorithms do not simplify or structure the input variety: they simply allow more operations to be performed with the help of a change of representation. Consequently, the degrees of the output polynomials or the size of the output matrix are not bounded by a constant and depend on the input's complexity. In geometrical frameworks, they are not meant to be used on the whole shape but rather on the the local patches such as the Bézier patches of a 3D model.

The algorithms (IM) and (Impl) both use interpolation technics in order to produce an implicit representation. They can be used in both situations, for generating a shape or obtaining an implicit representation of an existing one. However, both may fail to find an algebraic variety corresponding to the input: (IM) requires a support of the implicit polynomials and may end up interpolating only a few of the input points if a bad support is

provided; (Impl) may find implicit polynomials of analytic parameterizations (it implicitizes correctly the parameterization $(\cos(t), \sin(t))$ of the circle for instance) but fails when there is none.

The differences between the algorithms (MPU) and (Slim) mainly lie in the way to measure a shape's quality. (MPU) is designed to handle sharp features with precision and efficiency while (Slim) is designed to generate local areas fitting to the output shape (and also provide a multi-scale approximation). Both the expectations of the user and the kind of shape to approximate are relevant to the choice of the algorithm. On top of that, the internal structure of the output of these two algorithms are quite different: one produces an octree the leaves of which hold local implicit polynomials, the other produces a tree of increasingly smaller balls that lead to weighted combinations of local implicit polynomials. Ideally, a rendering engine flexible enough to allow different kind of implicit representations should be used, in order to use the advantages of each method on 3D models with different kinds of features. Another possibility is to mesh the shapes, standardising the representation's format.

(Geom) is more useful for shape simplification or shape repairing as it produces models made of very simple parts. Also, contrary to (MPU) and (Slim), (Geom) ignores straightforward the points that cannot be fitted to these simple geometric shapes. Thus, it can be used on point clouds distorted by a lot of noise of certain types (gaussian noises for instance) to filter out the noisy points. It is also very useful for splitting a complex model into small simple pieces, allowing a preprocessing of a point cloud for a more detailed implicitization algorithm or filling the defects that may appear when merging the local patches together.

There are several differences between the algorithms (CF), (GB) and (MQ). Firstly, the format of their output is not the same: (CF) only provides implicit equations, (GB) provides implicit equations that form a Gröbner basis of the variety V, which is a finer information as projections of $\mathcal{I}(V)$ over $\mathbb{K}[x_k, \ldots, x_n]$ are also computed in the process for example, and (MQ) provides an implicit matrix representation which is harder to manipulate but more compact. Secondly, the output format of (MQ) allows to solve the inversion problem: given a point P on the variety, we can compute the parameters t of the input parameterization ϕ such that $\phi(t) = P$ simply by computing the kernel of the output matrix evaluated at P. This is a very useful operation to have access to, as explained in the section 5.3.3. The matrix produced by the method (IM) shares that same property, even though it is much less efficient because the matrix is much bigger. Note that the GCD method described in the section 5.3.3 - which requires the computation of a μ-basis of a rational parameterization - may also be used to solve the inversion problem: the GCD computed for checking the ownership of a point outright gives the related parameters when the point belongs to the variety. Also, although they all handle varieties of any positive dimension embedded in spaces of any dimension, (CF) fares better on low dimensional varieties (eg. curves).

6.0.2 Performances: change of representation algorithms

The tests done for this section have been run using an Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz, 2400 Mhz, 2 Core(s), 4 Logical Processor(s) on a x64 machine with 8GB of RAM. We used Maple implementations of the algorithms (Impl), (CF) and (GB) and a SageMath implementation of (MQ).

One problem shared by many exact implicitization algorithms is the presence of base points in the given parameterization. As seen in the section 2.1, base points sometimes cannot be avoided and are an obstacle to even the simplest Gröbner basis methods. Base points are also troublesome in direct uses of resultants as they may cause a resultant to identically vanish depending on what that resultant is applied on. However, although it may require tricks to bypass the obstacle, most of algorithms (and all of those discussed in this section) can deal with parameterizations that have base points. To be exact, the algorithm presented in [6] that is similar to (MQ) requires a subtle hypothesis on the base points of the input: the parameterization must be a local complete intersection around its base points. However, as said in [6], the problem arising when we do not have a local complete intersection is not disastrous.

Note that the algorithm A.7, (CF), can be easily parallelised as the computation of each implicit equation can be done independently. This effectively reduces its runtime by a factor equal to the number of equations required (three for space curves). For a fair comparison, we do not use parallelisation in the runtimes presented here because we have not studied the potential of parallelisation of the other algorithms.

The output of the matrix representation (MQ) and interpolation (MI) algorithms are symbolic matrices that have the drop of rank property. In order to get outputs comparable to the outputs of (CF), (GB) and (Impl), it is possible to retrieve implicit equations from such matrices by computing their minors of size equal to the maximal rank of the matrix. However, this is not the aim of these algorithms, not to mention that computing all the minors quickly becomes too expensive. We could nonetheless compute the minors of the output of (MQ) for space curves of parametric degree up to 7. These minors yield, up to constant factors, the same equations as the (GB) method. The minors of the matrices computed by (MI) depend on the set of monomials used as input. Computing these minors takes longer than 30 min already for space curves of parametric degree 4. The algorithm (CF) outputs a different set of equations. For curves of moderate and high degrees, the Sylvester resultant computation represents more than 99% of (CF)'s computation time.

For curves parameterized by polynomials of degree up to 6, all methods are comparable in running time, with (CF) and (GB) being slightly better generically. For larger degrees (GB) outperform (CF) by a factor that depends on the degree and the sparseness of the parametric polynomials and ranges from 10 to 10^3. While (MQ) fares correctly for generic space curves of degree less than 10, it is never the fastest method.

In the table 6.2, we show the effect of the μ-basis degree on the algorithms' efficiency. Indeed, the degrees of the μ-basis influences the number and degrees of implicit equations of a variety. In that table, the space curves are generated by taking 12 random polynomials

degree of parameterization and μ-basis degrees	(Impl)	(CF)	(GB)	(MQ)
$\mathbf{7(2,2,3)}$	80ms 52 equations	120ms 3 equations	30ms 8 equations	80ms Matrix 3x7
$\mathbf{7(1,2,4)}$	120ms 77 equations	90ms 3 equations	30ms 6 equations	100ms Matrix 4x8
$\mathbf{9(3,3,3)}$	170ms 79 equations	190ms 3 equations	90ms 12 equations	50ms Matrix 3x9
$\mathbf{9(1,4,4)}$	310ms 163 equations	210ms 3 equations	80ms 10 equations	130ms Matrix 4x9
$\mathbf{9(1,1,7)}$	300ms 157 equations	240ms 3 equations	70ms 9 equations	290ms Matrix 7x14

Table 6.2: Runtime and output size of different algorithms depending on the μ-basis degrees of the parameterization

of fixed degrees $(p_{1w}, p_{1x}, p_{1y}, p_{1z}, p_{2w}, p_{2x}, p_{2y}, p_{2z}, p_{3w}, p_{3x}, p_{3y}, p_{3z})$ to be the μ-basis; the space curve's homogeneous parameterization is then given by:

$$\left(\begin{vmatrix} p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \\ p_{1z} & p_{2z} & p_{3z} \end{vmatrix} : - \begin{vmatrix} p_{1w} & p_{2w} & p_{3w} \\ p_{1y} & p_{2y} & p_{3y} \\ p_{1z} & p_{2z} & p_{3z} \end{vmatrix} : \begin{vmatrix} p_{1w} & p_{2w} & p_{3w} \\ p_{1x} & p_{2x} & p_{3x} \\ p_{1z} & p_{2z} & p_{3z} \end{vmatrix} : - \begin{vmatrix} p_{1w} & p_{2w} & p_{3w} \\ p_{1x} & p_{2x} & p_{3x} \\ p_{1y} & p_{2y} & p_{3y} \end{vmatrix} \right) \tag{6.1}$$

We see that (Impl) is much less efficient when μ_n and μ_{n-1} are higher (here, $n = 3$). Indeed, (Impl) requires the computation of implicit equations of higher degrees in that situation in order to implicitize the curve. Moreover, because an implicit equation of a given degree produces three other equations a degree further, (Impl) computes a lot of redundant equations. While the method (MQ) is also running worse when the μ-basis is unbalanced, it is not so clear for the method (CF). Indeed, the number of output equations is always 3 when using (CF) and the measure of the complexity really is the degree of the parameterization. The differences in (CF)'s runtime is unclear and may be due to the relative height of the parameterizations' coefficients because of the way we generate the parameterizations with the equations (6.1). The Gröbner basis algorithm (GB) fares the best regarding the μ-basis degrees as unbalanced degrees are even slightly easier to implicitize with that method. Indeed, the problem with Gröbner basis algorithms usually do not lie in the degree of the implicit equations sought but rather in polynomial gcd that needs to be computed during the process. The condition of genericity for these Gröbner basis algorithms to run fast is a technical condition: the variables x_i, t_i must be in *simultaneous Noether position* w.r.t. polynomials generating the ideal of which we compute the Gröbner basis (see [3]).

Though the runtime per equation of (Impl) is lower than that of (CF), the number of implicit equations that (Impl) outputs cannot be predicted as it does not depend on the input parameterization's degree only but also on the balance of its μ-basis (and the precision of (Impl)'s internal computations of integrals and kernel). Amongst these, some equations

can be redundant. In contrast, (CF) return the implicit equations one by one, and Theorem 12 guarantees that 3 of them define the curve set-theoretically. The algorithm (GB) also returns a small set of equations, though not necessarily minimal, and (MQ) produces a compact matrix.

On the other hand, (CF) always outputs equations of the same total degree which is the degree δ of the parameterization while (GB) and (MQ) produce equations of optimal or close-to-optimal degrees (see the section 5.2.3 for a discussion about non-optimal quadratic matrices). The total degree of equations outputted by (Impl) must be provided by the user. For generic space curves, the degree of the implicit equations can be as low as $\lceil \frac{2\delta}{3} \rceil$. However, for non-generic space curves (with unbalanced μ-basis degrees), some of the implicit equations may require to be of degree δ (this extreme case is met for space curves included in a plane, that is with $\mu_1 = 0$). In such situation, the runtime of (Impl) is very high because it outputs extremely many implicit equations of relatively high (most of them being redundant).

In terms of theoretical time complexity, the Gröbner basis algorithms are known to have an horrendous and rather misleading worst-case complexity because of the same reason mentioned above (described in [3]). This worst-case complexity cannot be lower than $O\left(n\delta\binom{n+d+\delta-1}{\delta}^{\omega}\right)$ asymptotically in δ, where we recall that n is the dimension of the ambient space, δ is the degree of the parameterization and d is the dimension of the variety V; ω is the matrix multiplication complexity exponent, which is $\omega \approx 2.373$ when using the Coppersmith–Winograd algorithm [46]. In generic cases though, this complexity can drop to $O\left(\delta^{3(d+n)}\right)$. The algorithm (CF) has at worst n times the complexity of resultant algorithms, namely a complexity of $O\left(n\delta^2\right)$ for Sylvester resultants (i.e. for the cases $d = 1$ and $n > 2$) and an asymptotic complexity of $O\left(n\delta^{O(d)}\right)$ for sparse resultants (see [23] for a more accurate complexity). The complexity of (MQ) for curves is the same as a $\left(\mu_n\binom{n+1}{2}\right) \times \left(\mu_n\binom{n+1}{2}\right)$-matrix kernel computation complexity (either exact or numerical depending on the use), so $O\left(\mu_n^{\omega} n^{2\omega}\right)$. Finally, (Impl) has a complexity of $O\left(\binom{n+k}{n}^2\right) \times O\left(\int \phi^{2k}\right)$ where k is the degree of the output equations provided by the user (typically, $\frac{d+1}{n}\delta \leq k \leq \delta$) and $O\left(\int \phi^{2k}\right)$ is the complexity of computing (numerically or exactly) an integral of type $\phi_1^{\alpha_1} \dots \phi_n^{\alpha_n}$ with $|\alpha| \leq 2k$.

7. CONCLUSION

We have seen that implicitization is not a trivial operation. Although most simple varieties, e.g. the hypersurfaces of degree 1 or 2, can be represented both implicitly and parametrically, and their representation can be switched effortlessly, it is not the case any more beyond that algebraic complexity threshold. For these difficult varieties, various implicitization algorithms exist without one being superior to all the others in all the situations. Simple varieties for their part, are often constructed as approximations of complicate shapes or point clouds and they also require different kinds of methods for which design purposes are at least as important as algorithmic efficiency.

The cases of plane curves, space curves and surfaces being the most useful geometrically, they are highlighted in our work. The implicitization of swept volumes are specific to 3D shapes (surfaces bounding an object). Most of the examples and applications found in the literature are restricted to the cases $n = 2$ or $n = 3$ and so are most of our examples too.

However, the algebra behind the methods that we developed are mostly better understood when dealing with the general case. Dealing with varieties of different dimensions embedded in spaces of different dimensions is one of the advantages of the algebraic theory underlying most of the algorithms presented in our work. Algebra also provides the correct ways to measure the inherent complexity of shapes. The degree of the varieties, of course, but also the algebraic genus for their rationality, the presence of base points and, as we have seen in Chapter 5, the balance of the μ-basis degrees. This latter algebraic invariant explains why varieties parameterized by low degree polynomials but unbalanced μ-basis degrees sometimes turn out to be more difficult to implicitize for several algorithms than varieties parameterized by polynomials of higher degrees.

When trying to generalise this syzygy-based quadratic matrix representation, the third syzygy module (that is, cubic relations between the parametric polynomials) has been considered. Surprisingly, it does not provide better results in the sense that we loose the equivalence of the drop of rank of the matrix evaluated at a point and the ownership of that point to the variety (there can be false positive). Another consideration is that looking at the theorems (20 and 21): $\mathbb{M}_\nu$ is a matrix representation for $\nu \geq \mu_n + \mu n - 1 - 1$, $\mathbb{MQ}_\nu$ is a matrix representation for $\nu \geq \mu_n - 1$…what would be the threshold for cubics? The answer is not clear as if such result exists, the threshold would surely involve μ_n and be lower than $\mu_n - 1$ somehow. Relations of degree 2 are thus special in that sense. As stated in Chapter 1, varieties that can be blown up to varieties of degree 2 are special in the sense that they are rational; we have no clue about a relation tying these two facts though.

The work done on swept volumes in Chapter 3 induces the question of whether algorithms should be developed to be more adapted to different kinds of informations that could be known about an object (in this situation, a swept volume is not just any kind of object but an object obtained from a rigid motion of a base volume, which can be seen as a generalisation of surfaces of revolution). Indeed, the difficulty of the problem of implicitization

may lead to try to integrate somehow the extraneous informations the user could know about the object in order to ease the problem. On the other hand, multiplying the number of representations may lead to compatibility problems: most probably, a balance between standardisation and adaptability is what we are heading to.

A continuation of the work done here would be to examine more closely the advantages and drawbacks of each implicitization algorithms and, more generally, algorithms for change of representations. This would not be an easy task because the perspective of implicitization algorithms may vary from one to another. An epitome of this fact can be seen in the various error measurements used when doing approximate implicitization: the MPU method uses a combination of the Taubin distance and the maximal angular distance of normals to spot edges and corners, the Slim method optimizes a regularized ℓ^2 norm weighted by a Gaussian-like weight, the geometric primitive extraction method used in Chapter 6 uses a fuzzy norm. Some error measurements are chosen in view of some design quality goal, some others are chosen for practical reasons of algorithm speed or convergence, most error measurements are chosen with a combination of design quality and practical reasons. It may be impossible to only choose an error measure that would be suitable for all the approximate implicitization algorithms. It might be possible to classify them in some wide classes of quality measurements, such as ℓ^1 distance (optimization of the homogeneous mean of errors), ℓ^∞ distance (optimization of the maximal error), with or without considering errors smoothness constraints, etc. The possibility to have multi-scale representations such as a detailed but slow representation and a fast but coarse representation is also to be considered, especially in situations of real-time rendering and for objects that can be previewed.

In terms of exact implicitization, the problem of finding the best algorithm to use in a given situation is slightly easier to grasp. The differences between algorithms lie in the types of variety handled, the input informations required by the algorithms, the efficiency of the algorithms and the format of the output implicit representation. In the industry, implicit representations are often implemented to be polynomials or functions; the implicit matrix representation that we developed in Chapter 5 may require updates of softwares for them to be usable, despite their advantages (in particular, the fact that they can solve the inversion problem, finding the parameters associated with a point lying on the variety). Aside from that practical issue, it should be possible to design a toolbox containing several implicitization algorithms and organise that toolbox so that the best algorithm is picked automatically in a given situation. This kind of work has been done before and the implicitization routines of the most complete softwares usually pick an algorithm amongst several ones depending on the complexity of the input. This has to be updated regularly, maybe including approximate implicitization methods in the toolbox.

Several ways to optimise existing algorithms can be considered. Parallelisation is an optimisation that is usually simple to implement. The algorithm of Chapter 4, for instance, can be parallelised easily to reduce its runtime by a factor of n since each equation is computed independently and only a very simple check should be performed on the randomised points in order to obtain a set of equations intersecting exactly on the variety that must be implicitized. Another optimisation, in particular when dealing with linear algebra

(which is often the case to some extent), is to make use of calculus in fields of non-zero characteristic. Indeed, computation in these fields is faster and, properly analysed, may provide the results of computation in field of characteristic zero. It is however a whole world in itself to study the relationship between fields of different characteristics and be able to use finite fields advantageously for speeding up computations.

A very trendy approach is the use of neural networks, and more precisely deep learning methods, to achieve various kinds of goal. Deep learning can indeed be trained on producing implicit representations starting from various kind of inputs. The results should be quite good for approximate implicitization but hardly usable for exact implicitization, as we leave the world of algebra. Another advantage of deep learning methods is that they get extremely efficient with the class of shapes they are trained on. While algebraic methods and, to a lesser extent, usual approximation methods are able to handle correctly a wide variety of shapes, even bizarre ones, because their heuristical aspects are adjusted by the implementations (e.g. the error measurement discussed before), neural networks automatically adapt to the objects they are given and are more efficient on similar ones.

However, we think that there is and there will always be room for algebraic standpoints when it comes to representations of geometric objects. Indeed, the richness of algebraic structures offers possibilities for geometric operations that are not possible otherwise, because polynomials are well-known and easy to handle. Approximate implicitization may have little use of algebraic theory; however, when it comes to exactness and conversions without loss of precision, the polymorphous algebra toolbox is there to avail.

ABBREVIATIONS - ACRONYMS

ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
NKUA	National and Kapodistrian University of Athens
INRIA	French National Institute for Research in Computer Science and Automation
ΑΠΘ	Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
AUT	Aristotle University of Thessaloniki
ΠΠ	Πανεπιστημίου Πατρών
UP	University of Patras
WLU	Wilfrid Laurier University of Waterloo
CAGD	Computer-Aided Geometric Design
CAE	Computer-Aided Engineering

APPENDIX A. ALGORITHMS

Figure A.1: Algorithm - Interpolation matrix

Input: V parameterized as in the equations (3).
Or a set of monomials S and a point cloud $(\overline{x}_k)_k \subset V$ of at least $|S|$ points such that S contains all monomials of a set of generators of $\mathcal{I}(V)$.

Output: An interpolation matrix $M(x)$ whose rank drops on V.

1: **if** S is not given **then**
2: Let $\nu \coloneqq \prod_{i=1}^{d} \max_j(\deg_{t_i}(f_j))$.
3: Let $S = (m_1, \ldots, m_{|S|})$ be the monomial basis of polynomials of degree $\leqslant \nu$.
4: **end if**
5: Pick μ random points $(\overline{x}_k)_{1 \leqslant k \leqslant \mu}$ on V with $|S| \leqslant \mu \leqslant 2|S|$.
6: Construct $M' \coloneqq (m_i(\overline{x}_k))_{1 \leqslant i \leqslant |S|, 1 \leqslant k \leqslant \mu}$.
7: Construct $M(x) \coloneqq \begin{pmatrix} M' \\ S(x) \end{pmatrix}$.

Figure A.2: Algorithm - MPU

Input: A point cloud $\mathcal{P}$.
The normals $\mathcal{N}$ of that point cloud.
Output: A suitable local implicit representation for $\mathcal{P}$.

Compute a bounding box A_1 of $\mathcal{P}$.
Rescale such that A_1 is a cube of diagonal length 1, i.e. of edge length $\frac{1}{\sqrt{3}}$.
Let $A = F = \emptyset$.
Let $S = \{A_1\}$.
while S is not empty **do**
    Pick $a \in S$; a is a cube of diagonal length d.
    Let C be the sphere centred on the cube a of radius $R = \alpha d$. (*typically,* $\alpha = 0.75$)
    if C contains less than N_{min} points **then** (*typically,* $N_{\text{min}} = 15$)
        Let C' be an enlargement of C that contains at least N_{min} points.
    end if
    Let f be a MPU local approximation of $\mathcal{P} \cap C'$. (*see A.3*)
    if it failed **then**
        Subdivide a into 8 cubes and add them to S.
        continue
    else if C contains no point **then**
        Add a to A and f to F.
        continue
    end if
    Let $\epsilon = \max_{p \in \mathcal{P} \cap C} |f(p)| / \|\nabla f(p)\|$.
    if $\epsilon < \epsilon_0$ **then** (*typically,* $\epsilon_0 = 10^{-4}$)
        Add a to A and f to F.
    else
        Subdivide a into 8 cubes and add them to S.
    end if
end while
return (A, F) after rescaling it back.

Figure A.3: Algorithm - Local MPU Approximation

Input: A sphere of centre c and radius R.
A cube a with the same centre c and diagonal length d.
A point cloud $\mathcal{P}' = (p_i)_i$ of at least N_{min} points inside that sphere.
The normals $\mathcal{N}' = (n_i)_i$ of that point cloud.
Output: A local approximation of $\mathcal{P}'$.

Let $n = \sum_i b\left(\frac{3\|p_i - c\|}{2R}\right) n_i$ where b is the quadratic B-Spline then normalise n.
Let θ be the maximal angle between n and $n_i \in \mathcal{N}'$.
if $|\mathcal{P}'| > 2N_{\text{min}}$ and $\theta \geq \pi/2$ **then** (*case (a)*)
    Let Q be the corners of a and its centre. ($|Q| = 9$)
    for $q \in Q$ **do**
        Get the 6 nearest neighbours $p^{(j)}$ of q in $\mathcal{P}'$. ($j = 1, \ldots, 6$)
        if $n^{(j)} \cdot \left(q - p^{(j)}\right)$ have different signs **then** Remove q from Q.
    end for
    if Q is empty **then return** FAIL.
    return f minimizing (3.1).
else if $|\mathcal{P}'| > 2N_{\text{min}}$ and $\theta < \pi/2$ **then** (*case (b)*)
    Let (u, v, n) be an orthonormal local coordinate system centred on c.
    return f of the form (3.2) minimizing $\sum_i w(p_i) f(p_i)^2$.
else (*case (c)*)
    Let $p^{(1)}, p^{(2)} \in \mathcal{P}'$ and θ such that $\theta = n^{(1)} \cdot n^{(2)} = \min_{i,j} n_i \cdot n_j$.
    if $\theta \geq \theta_{\text{sharp}}$ **then return** f of the form (b) (*typically,* $\theta_{sharp} = 0.9$)
    Split $\mathcal{P}' = \mathcal{P}'_1 \cup \mathcal{P}'_2$ using a spherical Voronoi partition w.r.t. $n^{(1)}$ and $n^{(2)}$. (*see [50]*)
    Let $e = n^{(1)} \times n^{(2)}$, an approximate of the direction of the potential edge.
    if $\max_i |n_i \cdot e| \leq \theta_{\text{corner}}$ **then** (*typically,* $\theta_{corner} = 0.7$)
        Let f_1, f_2 of the form (b) w.r.t. $\mathcal{P}'_1, \mathcal{P}'_2$ respectively.
        return $f = \min(f_1, f_2)$.
    end if
    for $p_i \in \mathcal{P}'$ **do**
        if $|n^{(1)} \cdot n_i| < |e \cdot n_i|$ and $|n^{(2)} \cdot n_i| < |e \cdot n_i|$ **then**
            Add p_i to a third set $\mathcal{P}'_3$ and remove it from $\mathcal{P}'_1$ or $\mathcal{P}'_2$.
        end if
    end for
    Let $p^{(3)}, p^{(4)} \in \mathcal{P}'_3$ such that $n^{(3)} \cdot n^{(4)}$ is the smallest amongst points in $\mathcal{P}'_3$.
    if $n^{(3)} \cdot n^{(4)} \geq \theta_{\text{sharp}}$ **then**
        Let f_1, f_2, f_3 of the form (b) w.r.t. $\mathcal{P}'_1, \mathcal{P}'_2, \mathcal{P}'_3$ respectively.
        return $f = \min(f_1, f_2, f_3)$.
    end if
    Split $\mathcal{P}'_3 = \mathcal{P}'_4 \cup \mathcal{P}'_5$ using a spherical Voronoi partition w.r.t. $n^{(3)}$ and $n^{(4)}$.
    Let f_1, f_2, f_4, f_5 of the form (b) w.r.t. $\mathcal{P}'_1, \mathcal{P}'_2, \mathcal{P}'_4, \mathcal{P}'_5$ respectively.
    return $f = \min(f_1, f_2, f_4, f_5)$.
end if

Figure A.4: Algorithm - Slim

Input: A point cloud $\mathcal{P}$.
The normals $\mathcal{N}$ of that point cloud.
Output: A suitable local implicit representation for $\mathcal{P}$.

Let Rep $= \emptyset$.
Let $\mathcal{B}_0 = \{B_{00}, B_{01}, \dots\}$ be a cover of $\mathcal{P}$ by balls B_{0i} of radius ρ_0.
(*typically, ρ_0 is 1/10 of the main diagonal of the whole object's bounding box*)
Let $U = \mathcal{P}$, a list of "uncovered" points.
Let $k = 1$ and $\rho_1 = g\rho_0$. (*typically, $g = \frac{\sqrt{5}-1}{2}$, the golden ratio conjugate*)
while U is not empty **do**
    Let $\rho_{k+1} = g\rho_k$.
    Let $\mathcal{B}_k = \{B_{k0}, B_{k1}, \dots\}$ be a cover of U by balls of radius ρ_k.
    for $B \in \mathcal{B}_k$ **do**
        Compute an approximation F_B by minimizing (3.4). (*typically, $T_{MDL} = 0.02$*)
        if $E(\rho_{k+1}) > E(\rho_k) < E(\rho_{k-1})$ and $\epsilon(\rho_{k+1}) < \epsilon(\rho_k) < \epsilon(\rho_{k-1})$ **then** (*see (3.5)*)
            Remove the points $\mathcal{P} \cap B$ from U and add (B, F_B) to Rep.
        else
            Optionally store (B, F_B) at the level k of a multi-scale representation.
        end if
    end for
    Increment k.
end while
return the representation Rep.

Figure A.5: Algorithm - Implicit representation of swept volume from implicit representation of base volume

Input: A base volume $\mathcal{B}$, possibly given with distance functions.
A rigid transformation $\mathcal{T}$.

Output: A procedural implicit representation of $\mathcal{T}(\mathcal{B})$, possibly allowing distance computation.

`/* Computing a bounding box can be done by` $\min_x := r + \min_t(v(t)_x)$`, etc. where` r `is the radius of a bounding sphere of` $\mathcal{B}$ `and` v `is the translation vector of` $\mathcal{T}$ `*/`

1: Compute a bounding box $\mathbb{B}$ of $\mathcal{T}(\mathcal{B})$.
`/* Compute a suitable partition of` $\mathbb{B}$ `*/`
2: Let $C := \{\mathbb{B}\}$
3: **for** $i = 0, \ldots, M$ **do**
4: Split C at the position $[\mathcal{T}(a + i(b-a)/M)](\text{Centre}(\mathcal{B}))$ w.r.t. the coordinate maximizing $|v'(a + i(b-a)/M)|$
5: **end for**
`/* Setup the tree structure of the representation */`
6: **for** $c \in C$ **do** (*see the remark 1 for a smarter loop*)
`/* Computing the intersection of a moving area with a cuboid. The formulae depend on the type of the moving area (cube, sphere…).*/`
7: Compute the local areas $\mathcal{A}_c := \{(A_i, [t_0, t_1]) \mid \forall t \in [t_0, t_1], c \cap [\mathcal{T}(t)](A_i) \neq \emptyset\}$
8: **end for**
9: **return** $C, \{\mathcal{A}_c\}$

Figure A.6: Algorithm - Usage of the swept volume implicit representation provided by the algorithm A.5

Usage: Check ownership of a query point P.
Compute intersections with a query ray R.

`/* Ownership of` P `*/`
1: Find cell $c \in C$ such that $P \in c$
2: **if** there is no such cell **then**
3: **return** false, "P is far away"
4: **end if**
5: Let $d \leftarrow +\infty$
6: **for all** $(A_i, [t_0, t_1]) \in \mathcal{A}_c$ **do**
7: Solve $(t, d_{\text{tmp}}) \leftarrow \min_{t \in [t_0,t_1]}(F_i([\mathcal{T}(t)]^{-1}(P)))$ (*)
8: **if** $[\mathcal{T}(t)]^{-1}(P)$ is on the inner boundary of A_i **then**
9: **return** true, "P is far inside"
10: **end if**
11: Let $d \leftarrow \min(d, d_{\text{tmp}})$
12: **end for**
13: **return** $d \leq 0, d$ (d *is a signed distance of* $\mathcal{T}(\mathcal{B})$*, assuming* F_i *are local signed distances of* $\mathcal{B}$)

`/* Intersection with ray` $R = \{R_o + sR_d \mid s \in \mathbb{R}_+\}$ `*/`
1: Find cells C_R such that $R \cap c \neq \emptyset, \forall c \in C_R$
2: Sort C_R by distance w.r.t. R_o
3: **for all** $c \in C_R$ **do**
4: **for all** $(A_i, [t_0, t_1]) \in \mathcal{A}_c$ **do**
5: Let $I \leftarrow \{t \in [t_0, t_1] \mid [\mathcal{T}(t)]^{-1}(R) \cap A_i \neq \emptyset\}$ (**)
6: **if** $I \neq \emptyset$ **then**
7: Let $s \leftarrow \min(\{s \mid t \in I, [\mathcal{T}(t)]^{-1}(R(s)) \in A_i, F_i([\mathcal{T}(t)]^{-1}(R(s))) \leq 0\})$ (*)
8: **return** $R(s)$
9: **end if**
10: **end for**
11: **end for**
12: **return** "R does not intersect $\mathcal{T}(\mathcal{B})$"

(*) Using Newton or bisection algorithms depending on the properties of F_i
(**) Using the Newton algorithm if a suitable ray/object distance is provided or the bisection algorithm else

Figure A.7: Implicit representations of a d-dimensional variety $V \subset \mathbb{P}^n$

Input: V, parameterized by $x_j = f_j(t), j = 0, \ldots, n$.
Number of iterations $\rho\,(= 3$ when $n = 3)$.
Output: ρ polynomials vanishing on V.

1: **for** $k = 1, \ldots, \rho$ **do**
2: Define a $(n - d - 1)$-dimensional linear subspace by affinely independent random points $\mathcal{G} = \{G_1, \ldots, G_{n-d-1}\}$ none $G_k \notin V$ $(1 \leqslant j \leqslant n - d - 1)$, and consider symbolic point $\xi = (\xi_0, \ldots, \xi_n)$.
3: Define $d+1$ hyperplanes H_i through $\mathcal{G}, \xi, P_i$, for random point sets P_i affinely independent from points in $\mathcal{G}$.
4: Set $x_j = f_j(t)$ in the equations of H_i: their resultant $R_\mathcal{G}$, where we eliminate t, is the sought polynomial in ξ.
5: Compute the extraneous factor E using the exterior algebra recursive formula. Then divide $R_\mathcal{G}$ by E as many times as possible to obtain $Eq_k := \mathcal{S}_V^\mathcal{G}$.
6: **end for**
7: **return** $Eq_1, \ldots, Eq_\rho$

Figure A.8: Algorithm - Construction of the matrices $\mathbb{MQ}_\nu$

Input: A parameterization ϕ of a curve as defined in (5.5).
An integer ν.
Output: A matrix $\mathbb{MQ}_\nu$.

1: Compute a basis of the moving hyperplanes following ϕ of degree ν and build the matrix $\mathbb{M}_\nu$.
2: Compute a basis $\langle Q_1, \ldots, Q_{c_\nu} \rangle$ of the vector space $W_\nu \,/\, V_\nu$; its k-th element is of the form

$$Q_k = \sum_{0 \leq i \leq j \leq n} \sum_{l=0}^{\nu} c_{k,l,i,j} s^{\nu-l} t^l x_i x_j$$

3: Define the matrices $M_{i,j} = (c_{k,l,i,j})_{l,k}$ and the matrix $\mathbb{Q}_\nu = \sum_{0 \leq i \leq j \leq n} M_{i,j} x_i x_j$.
4: Return the concatenated matrix

$$\mathbb{MQ}_\nu = \left(\begin{array}{c|c} \mathbb{M}_\nu & \mathbb{Q}_\nu \end{array} \right).$$

Figure A.9: Algorithm - Construction of $\mathbb{MQ}_{\mu_n-1}$

Input: A parametric curve ϕ defined by (5.5).
Output: The matrix $\mathbb{MQ}_{\mu_n-1}$.

1: Compute a μ-basis $(p_1, \ldots, p_n)$ of ϕ. Let μ_i be the degree of p_i and assume that $\mu_1 \leq \cdots \leq \mu_n$.
2: Let $\mathcal{B}$ be a basis of the polynomial of degree $\mu_n - 1$, for instance

$$\mathcal{B} := \{s^{\mu_n-1}, s^{\mu_n-2}t, \ldots, t^{\mu_n-1}\}.$$

3: Initialize the matrix $\mathbb{MQ}_{\mu_n-1}$ to the empty matrix. We build it by successively adding columns as follows.
4: For i from 1 to $n-1$ add a block of $\mu_n - \mu_i$ columns to the matrix $\mathbb{MQ}_{\mu_n-1}$ corresponding to the coefficients of the polynomials

$$\{s^{\mu_n-\mu_i-1}p_i, s^{\mu_n-\mu_i-2}tp_i, \ldots, t^{\mu_n-\mu_i-1}p_i\}$$

with respect to the polynomial basis $\mathcal{B}$.
5: **for** $i = 1, \ldots, n-1$ **do**
6: **for** $j = i+1, \ldots, n$ **do**
7: if $\nu_{i,j} := \mu_i + \mu_j - \mu_n - 1 \geq 0$ then add a block of $\nu_{i,j} + 1$ columns to the matrix $\mathbb{MQ}_{\mu_n-1}$ corresponding to the coefficients of the Sylvester forms

$$\{\mathsf{Syl}_\alpha(p_i, p_j) : |\alpha| = \nu_{i,j}\}$$

with respect to the polynomial basis $\mathcal{B}$.
8: **end for**
9: **end for**
10: **return** Return the matrix $\mathbb{MQ}_{\mu_n-1}$.

REFERENCES

[1] Pierre Alliez, David Cohen-Steiner, Yiying Tong, and Mathieu Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 39–48, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[2] Chanderjit Bajaj and Insung Ihm. Hermite interpolation of rational space curves using real algebraic surfaces. In *Proc. ACM Symp. on Comput. Geometry*, pages 94–103, 1989.

[3] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of the F5 Gröbner basis algorithm. *Journal of Symbolic Computation*, 70:49–70, September 2015.

[4] David N. Bernshtein. The number of roots of a system of equations. *Functional Analysis and Its Applications*, 9(3):183–185, 1979.

[5] Bruno Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Dissertation an dem Math. Inst. der Universität von Innsbruck, 1965.

[6] Laurent Busé. Implicit matrix representations of rational Bézier curves and surfaces. *Computer-Aided Design*, 46:14–24, Jan 2014. Spec. Issue 2013 SIAM Conf. Geometric & Physical Modeling.

[7] Laurent Busé and Marc Chardin. Implicitizing rational hypersurfaces using approximation complexes. *J. Symbolic Comput.*, 40(4-5):1150–1168, 2005.

[8] Laurent Busé and Thang Luu Ba. Matrix-based implicit representations of rational algebraic curves and applications. *J. CAGD*, 27(9):681–699, 2010.

[9] Laurent Busé, André Galligo, and Jiajun Zhang. Extraction of cylinders and cones from minimal point sets. *Graphical Models*, 86:1–12, Jul 2016.

[10] Laurent Busé, Clément Laroche, and Fatmanur Yıldırım. Implicitizing rational curves by the method of moving quadrics. *Computer-Aided Design*, 114:101–111, Sep 2019.

[11] Falai Chen and Wenping Wang. The μ-basis of a planar rational curve - properties and computation. *Graphical Models*, 64:368–381, 02 2003.

[12] Robert M. Corless, Mark W. Giesbrecht, Ilias S. Kotsireas, and Stephen M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In *Proc. AISC*, pages 174–183, 2000.

[13] David A. Cox. Bezoutians and Tate resolutions. *J. Algebra*, 311(2):606–618, 2007.

[14] David A. Cox, John Little, and Donal O'Shea. *Using algebraic geometry*. Number 185 in GTM. Springer-Verlag, New York, 2nd edition, 2005.

[15] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Springer International Publishing, 2015.

[16] David A. Cox, Thomas W. Sederberg, and Falai Chen. The moving line ideal basis of planar rational curves. *J. CAGD*, 15(8):803–827, 1998.

[17] John Dalbec and Bernd Sturmfels. Introduction to Chow forms. In Neil L. White, editor, *Invariant Methods in Discrete and Computational Geometry: Proc. Curaçao Conference, 13–17 June, 1994*, pages 37–58. Springer, 1995.

[18] Carlos D'Andrea. Macaulay-style formulas for sparse resultants. *Trans. Amer. Math. Soc.*, 354:2595–2629, 2002.

[19] Carlos D'Andrea. On the structure of μ-classes. *Communications in Algebra*, 32:159–165, 03 2004.

[20] Gema M. Diaz-Toca and Laureano Gonzalez-Vega. Barnett's theorems about the greatest common divisor of several univariate polynomials through Bézout-like matrices. *J. Symbolic Comput.*, 34(1):59–81, 2002.

[21] Tor Dokken. Approximate implicitization. In *Mathematical methods for curves and surfaces (Oslo 2000)*, Innov. Appl. Math., pages 81–102. Vanderbilt Univ. Press, Nashville, 2001.

[22] David Eisenbud. *Commutative algebra*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. With a view toward algebraic geometry.

[23] Ioannis Z. Emiris. On the complexity of sparse elimination. *Journal of Complexity*, 12(2):134–166, jun 1996.

[24] Ioannis Z. Emiris and John F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *Journal of Symbolic Computation*, 20(2):117–149, Aug 1995.

[25] Ioannis Z. Emiris, Vissarion Fisikopoulos, Christos Konaxis, and Luis Peñaranda. An oracle-based, output-sensitive algorithm for projections of resultant polytopes. *International Journal of Computational Geometry & Applications*, 23(04n05):397–423, Aug 2014.

[26] Ioannis Z. Emiris, Tatjana Kalinka, and Christos Konaxis. Geometric operations using sparse interpolation matrices. *Graphical Models*, 82:99–109, November 2015.

[27] Ioannis Z. Emiris, Tatjana Kalinka, Christos Konaxis, and Thang Luu Ba. Sparse implicitization by interpolation: characterizing non-exactness, and an application to computing discriminants. *J. CAD*, 45:252–261, 2013.

[28] Ioannis Z. Emiris, Christos Konaxis, Ilias S. Kotsireas, and Clément Laroche. Matrix representations by means of interpolation. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 149–156, New York, NY, USA, 2017. ACM.

[29] Ioannis Z. Emiris, Christos Konaxis, and Clément Laroche. Implicit representations of high-codimension varieties. *Computer Aided Geometric Design*, 74:101764, Oct 2019.

[30] Elisabetta Fortuna, Patrizia Gianni, and Barry Trager. Generators of the ideal of an algebraic space curve. *Journal of Symbolic Computation*, 44(9):1234–1254, Sep 2009.

[31] Israel M. Gelfand, Mikhail Kapranov, and Andrei Zelevinsky. *Discriminants, resultants and multidimensional determinants*. Birkhäuser, Boston, 1994.

[32] Herwig Hauser. The Hironaka theorem on resolution of singularities. *Bulletin of the American Mathematical Society*, 40(03):323–404, May 2003.

[33] Francis Hill. *Computer graphics : using OpenGL*. Pearson Prentice Hall, Upper Saddle River, NJ, 2007.

[34] Heisuke Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero. *The Annals of Mathematics*, 79(1):109, Jan 1964.

[35] Friedrich Hirzebruch. *Topological methods in algebraic geometry*. Springer Berlin Heidelberg, 1966.

[36] Hoon Hong, Zachary Hough, and Irina A. Kogan. Algorithm for computing μ-bases of univariate polynomials. *journal of Symbolic Computation*, 80:844–874, 2017.

[37] Gabriela Jeronimo, Teresa Krick, Juan Sabia, and Martín Sombra. The computational complexity of the Chow form. *Foundations of Computational Mathematics*, 4(1):41–117, Feb 2004.

[38] Xiaohong Jia, Xiaoran Shi, and Falai Chen. Survey on the theory and applications of μ-bases for rational curves and surfaces. *journal of Computational and Applied Mathematics*, 329:2–23, 2018. The International Conference on Information and Computational Science, 2–6 Aug. 2016, Dalian, China.

[39] Xiaohong Jia, Haohao Wang, and Ron Goldman. Set-theoretic generators of rational space curves. *J. Symbolic Comput.*, 45(4):414–433, 2010.

[40] Jean-Pierre Jouanolou. Formes d'inertie et résultant: un formulaire. *Advances in Mathematics*, 126(2):119–250, 1997.

[41] Jean-Pierre Jouanolou. An explicit duality for quasi-homogeneous ideals. *J. Symbolic Comput.*, 44(7):864–871, 2009.

[42] Young J. Kim, Gokul Varadhan, Ming C. Lin, and Dinesh Manocha. Fast swept volume approximation of complex polyhedral models. *Computer-Aided Design*, 36(11):1013 – 1027, sep 2004. Solid Modeling Theory and Applications.

[43] Leif Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*. ACM Press, January 2001.

[44] Ernst Kunz. *Introduction to commutative algebra and algebraic geometry*. Modern Birkhäuser Classics. Birkhäuser Basel, 2013.

[45] Clément Laroche. An implicit representation of swept volumes based on local shapes and movements. arXiv technical report, 2020. `https://arxiv.org/abs/2003.11527`.

[46] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation - ISSAC '14*. ACM Press, 2014.

[47] William E. Lorensen and Harvey E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, Aug 1987.

[48] Francis Sowerby Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, s1-35(1):3–27, May 1902.

[49] Thom Mulders and Arne Storjohann. On lattice reduction for polynomial matrices. *journal of Symbolic Computation*, 35(4):377–401, 2003.

[50] Hyeon-Suk Na, Chung-Nim Lee, and Otfried Cheong. Voronoi diagrams on the sphere. *Computational Geometry*, 23(2):183–194, Sep 2002.

[51] Vincent Neiger and Vu Thi Xuan. Computing canonical bases of modules of univariate relations. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 357–364, New York, NY, USA, 2017. ACM.

[52] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Trans. Graph.*, 22(3):463–470, July 2003.

[53] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Mathieu Desbrun, and Helmut Pottman. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. *Eurographics Symposium on Geometry Processing 2005, Eurographics Association, 149-158 (2005)*, January 2005.

[54] Arno Pauly. On the topological aspects of the theory of represented spaces. *arXiv e-prints*, Apr 2012.

[55] Arno Pauly and Florian Steinberg. Comparing representations for function spaces in computable analysis. *CoRR*, abs/1512.03024, 2015.

[56] Nicolas Perrin, Olivier Stasse, Léo Baudouin, Florent Lamiraux, and Eiichi Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics*, 28(2):427–439, March 2012.

[57] Sonia L. Rueda, Juana Sendra, and J. Rafael Sendra. An algorithm to parametrize approximately space curves. *J. Symbolic Computation*, 56:80–06, 2013.

[58] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[59] Thomas W. Sederberg and Falai Chen. Implicitization using moving curves and surfaces. In R. Cook, editor, *Proc. SIGGRAPH*, volume 29, pages 301–308. Addison Wesley, 1995.

[60] Thomas W. Sederberg, Ron Goldman, and Hang Du. Implicitizing rational curves by the method of moving algebraic curves. *J. Symbolic Comput.*, 23(2-3):153–175, 1997.

[61] J. Rafael Sendra, David Sevilla, and Carlos Villarino. Algebraic and algorithmic aspects of radical parametrizations. *Computer Aided Geometric Design*, 55:1–14, Jul 2017.

[62] Igor R. Shafarevich. *Basic algebraic geometry*, volume 1. Springer, 2013.

[63] Igor R. Shafarevich and Alexey O. Remizov. *Linear algebra and geometry*. Springer, New York, 2012.

[64] Ning Song and Ron Goldman. μ-bases for polynomial systems in one variable. *Computer Aided Geometric Design*, 26(2):217–230, 2009.

[65] Giovanni Staglianò. Macaulay2 package “Resultants”, May 2018. Available at `http://www2.macaulay2.com/Macaulay2/doc/Macaulay2-1.14/share/doc/Macaulay2/Resultants/html/`.

[66] Bernd Sturmfels. *Algorithms in invariant theory*. Texts and Monographs in Symbolic Computation. Springer, 2008.

[67] James Joseph Sylvester. A method of determining by mere inspection the derivatives from two equations of any degree. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 16(101):132–135, Feb 1840.

[68] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13:1115–1138, December 1991.

[69] Holger Täubig, Berthold Bäuml, and Udo Frese. Real-time swept volume and distance computation for self collision detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1585–1592. IEEE, Sep. 2011.

[70] Gilles Villard. Computing popov and hermite forms of polynomial matrices. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC ’96, pages 250–258, New York, NY, USA, 1996. ACM.

[71] Haohao Wang, Xiaohong Jia, and Ron Goldman. Axial moving planes and singularities of rational space curves. *Computer Aided Geometric Design*, 26(3):300–316, 2009.

[72] Xinyu Zhang, Young Kim, and Dinesh Manocha. Reliable sweeps. In *2009 SIAM/ACM Joint Conference on Geometric and Physical Modeling on - SPM ’09*, pages 373–378. ACM Press, Jan 2009.

[73] Wei Zhou, George Labahn, and Arne Storjohann. Computing minimal nullspace bases. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, ISSAC ’12, pages 366–373, New York, NY, USA, 2012. ACM.

[74] Matthias Zwicker and Craig Gotsman. Meshing point clouds using spherical parameterization, 2004.