Towards an Integrated Development tool for GA and a symbolic CGA implementation based on CasADi for application in robotics.

Oliver Rettig, Fabian Hinderer, and Marcus Strand

Robot-and-Human-Motion-Lab (RaHM-Lab), DHBW-Karlsruhe
(Baden-Wuerttemberg Cooperative State University), Germany
Oliver.Rettig@dhbw-karlsruhe.de
`https://www.karlsruhe.dhbw.de/rahmlab`

Summary of the Abstract

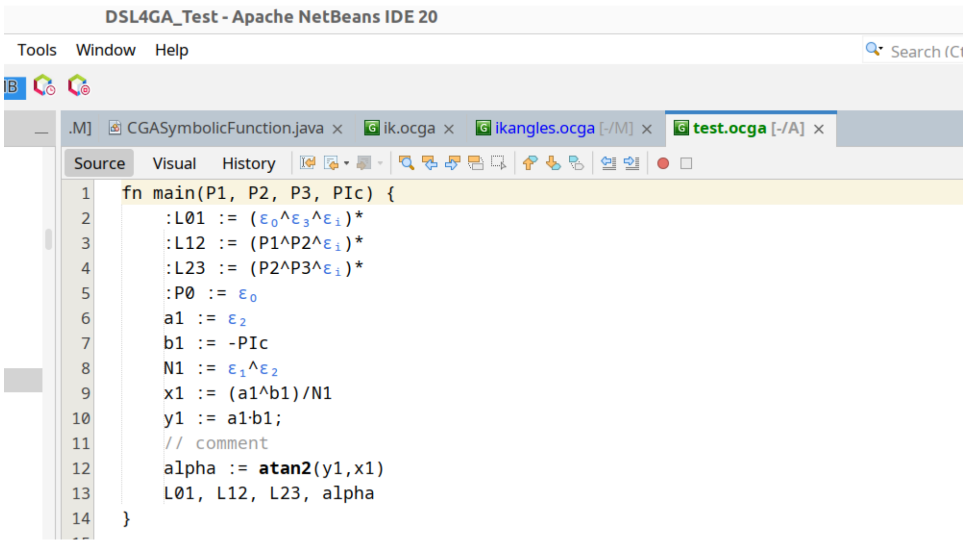
With *CasADi*[1] a framework for nonlinear optimization, optimal control and model predictive control is available and very popular in robotics research and applications. It employs a very fast acyclic graph modeling structure. The aim of this project is to simplify modelling with geometric algebra (GA) not only in the context of robotics by providing a GA library based on *CasADi* and by full GA tool integration into the *Apache-Netbeans IDE*[2]. The latter is an integrated development environment with syntax-highlighting and debugging functionality. The conference contribution will be to present and discuss the current state of the project and the direction of further developments.

Introduction

There is increased interest in usage of geometric algebra for modelling and implementing algorithms in robotics as can be seen in the number of publications about applications e.g. listed in [4]. Their applications range from DH-parameters determination [3] and symbolic implementations of inverse kinematics algorithms, e.g. [9] and ends up in complex algorithms for solving optimal control problems [11] or identification of singularities [10]. With Garamon[3] or *Gaalop*[4] generators for fast GA libraries are available and with *Gaalop*[3] and *Ganja.js*[5] there are also web-based tools available which facilitates a lot experimenting and starting to learn geometric algebra.

The aim of this project is to bring together fast GA implementations for real world robotics applications and the need for simplification the process of modelling with GA not only for newcomers in GA. To do this the development

[1] `https://web.casadi.org/`
[2] `https://netbeans.apache.org`
[3] `https://github.com/vincentnozick/garamon`
[4] `http://www.gaalop.de/`
[5] `https://enkimute.github.io/ganja.js`



```
fn main(P1, P2, P3, PIc) {
    :L01 := (ε₀^ε₃^εᵢ)*
    :L12 := (P1^P2^εᵢ)*
    :L23 := (P2^P3^εᵢ)*
    :P0 := ε₀
    a1 := ε₂
    b1 := -PIc
    N1 := ε₁^ε₂
    x1 := (a1^b1)/N1
    y1 := a1·b1;
    // comment
    alpha := atan2(y1,x1)
    L01, L12, L23, alpha
}
```

Figure 1: Geometric algebra inside the Netbeans IDE.

of a new domain specific language for geometric algebra (GA DSL) is started [3] and integrated into the Apache Netbeans IDE[2] combined with a fast implementation based on *CasADi*[1] a popular libaray in the field of robotics. This makes calculation of the Jacobian and Hessian matrices available by automatic differentiation.

The conference contribution will be to present and discuss the current state of the project and the direction of further developments.

Software Architecture

The idea of the software architecture is to create a modular framework, which allows to combine components to work with GA based on plain *Java* alone or with support of the Apache-Netbeans-IDE.

Furthermore it supports a new GA DSL, which is isolated from the GA implementation allowing to switch between different GA libraries. With *GACalcAPI*[6] a service provider interface (SPI) is established to integrate symbolic GA implementations. Its default implementation *CGACasADi*[7] allows to generate c-code without any dependencies to any library. Thus it is possible to use generated implementations of GA based algorithms outside the *Java* world.

With *Euclid3DViewAPI*[8] a SPI is provided to integrate 3D visualisation tools. *Euclid3D*[9] is its default implementation based on *JZY3D*[10]. This implementation allows to visualize simple geometric objects like points, lines, planes, spheres ... together with complex objects defined by shapes read from external

[6] `https://github.com/orat/GACalcAPI`
[7] `https://github.com/orat/CGACasADi`
[8] `https://github.com/orat/Euclid3DViewAPI`
[9] `https://github.com/orat/EuclidView3d`
[10] `http://www.jzy3d.org/`

files like robots. *Euclid3D* is not fast enough to visualize simulations due to some limitations of *JZY3D*. That's why it is planed to integrate external visualizers like the one from *ganja.js* or *Webots*[11] the new default one from *ROS2*[12] the robot operating system which is very popular in the scientific community of robotics research.

A domain specific language for GA (GA DSL)

Different to the operator syntax of the script languages (e.g. *CluScript*[13], *GaalopScript*) *Unicode* additional to simple ASCII characters symbols (see tab. 1) are used. This improves readability. The geometric product (gp) is represented by a simple space character. This is common in mathematical text books but an implementation of a corresponding syntax parser has turned out as much more complex than the usage of a printable symbol as done by existing geometric algebra scripting languages. Functions can be defined in the DSL itself to extend the build-in ones. Further language structures and especially a GA specific type system which allows to define subtypes of multivectors inside the DSL itself and structures to work with matrices of multivectors are planned.

Implementation of a generator to build an optimized calculation graph

A *CasADi* (acyclic) graph is build up automatically from the geometric algebra expressions formulated in the GA DSL at compile time. The generator (in the sense of [13]) *CGACasADi*[7] is implemented in *Java* and it uses *JCasADi*[1] a *Java*-Wrapper for *CasADi*, which we have build with help of *Swig*[14].

The gp is based on a geometric model specific cayley-table formulation. The further products (left- and right contraction the outer-, inner-, dot- and scalar-product (definition following [7]) are based on the gp and a grade selection operator. The linear operators dual/undual, reverse, conjugate and grade involution are implemented model free based on matrix formulations. Non linear operators are implemented symbolically and specific for the conformal model only. Exponentiation is implemented without usage of taylor series following [5] and [6] for bivectors only. Normalization and square roots are implemented following [2]. The inverse and constitutive the (right) division operator are implemented following [1]. Additional, for versor only arguments, a generic model free implementation is provided.

Implementation of a *Java*-Wrapper to use *CasADi*

CasADi is a fairly huge library and written in *C++*. To use in *Java* a way to call functions of compiled native libraries is needed.

The mapping between languages itself is non-trivial due to the differences between them. This spans from syntactical properties like multiple inheritance and operator overloading, which both is supported in C*C++* and not in *Java*, to subtle implementation details like that a *C++* template definition won't be

[11] `https://cyberbotics.com/` [12] `https://docs.ros.org/en/rolling/`
[13] `https://clucalc.software.informer.com` [14] `https://www.swig.org/`

Precedence	Symbol	Unicode code-points	Description
	$dot(\mathbf{X})$		dot product
	$scp(\mathbf{X})$		scalar product
	$negate14(\mathbf{X})$		neg. 1. and 4. grade
6	$\mathbf{X}^{*}$	\u002A	dual
6	$\mathbf{X}^{-*}$	\u207B \u002A	undual
6	$\mathbf{X}^{\sim}$	\u02DC	reverse
6	$\mathbf{X}^{\dagger}$	\u2020	Clifford conjugate
6	$\mathbf{X}^{\wedge}$	\u005E	grade involution
5	$-\mathbf{X}$	\u002D	negate
4	(space)	\u0020	geometric product
3	$\wedge$	\u2227	outer product
3	$\vee$	\u2228	regressive product
3	$\rfloor$	\u230B	left contraction
3	$\lfloor$	\u230A	right contraction
3	$\cdot$	\u22C5	inner product
1	$+$	\u002B	sum
1	$-$	\u002D	difference
	$\langle X \rangle_{\mathrm{p}}$	$\langle$= \u003C, $\rangle$= \u003E	extraction of grade p
	$normalize(\mathbf{X})$		(euclidean) normalize
	$norm(\mathbf{X})$		euclidean norm
	$sqrt(\mathbf{X})$		square root
	$exp(\mathbf{X})$		exponentiation
	$atan2(\mathbf{X})$		atan2 (for scalar only)
6	$\mathbf{X}^{-1}$	\u207B \u00B9	inverse
2	$/$	\u002F	right division (inv. gp)
6	$\mathbf{X}^{2}$	\u00B2	square
3	$\cap$	\u2229	meet
3	$\cup$	\u222A	join
	ϵ_0	\u03B5 \u2080	origin $\langle\epsilon_0\rangle$
	ϵ_1	\u03B5 \u2081	x $\langle\epsilon_1\rangle$
	ϵ_2	\u03B5 \u2082	y $\langle\epsilon_2\rangle$
	ϵ_3	\u03B5 \u2083	z $\langle\epsilon_3\rangle$
	ϵ_i	\u03B5 \u1D62	infinity$\langle\epsilon_i\rangle$
	ϵ_+	\u03B5 \u208A	$\epsilon_0 + \frac{1}{2}\epsilon_i$
	ϵ_-	\u03B5 \u208B	$\frac{1}{2}\epsilon_i - \epsilon_0$
	π	\u03C0	3.1415...

Table 1: 1. Linear- , 2. non-linear operators/functions, 3. predefined symbols: The given *Unicode* code-points are used in the GA DSL to get the shown symbols. A higher precedence number results in a higher binding strength.

compiled into the binary or that *C++* uses monomorphization for template instances while *Java* uses type erasure for its generics. JNI the standard way to implement a wrapper around native code by manually programming is error-prone, cumbersome and time-consuming. That's why we utilize *Swig*[14] to create glue code semi-automatically. *Swig* provides a lot of support doing the mapping, however it can still be complex and time consuming to create a working and correct mapping. *CasADi* uses multiple *C++* specific features in its public API. And although it provides a *Swig* file aimed at Python, Matlab and Octave, this was not of much use due to being not easily comprehensible and not very much suitable for a statically typed language like *Java*.

JCasADi[15] is our-*Java* wrapper of *CasADi*. Primarily due to the lack of operator overloading in *Java* it is slightly more laborious to be used directly in comparison to the Python version. However its usage is hidden for the end user, has been proven useful in the implementation of *CGACasADi*[7] and works stable so far.

Integration into the Apache-Netbeans-IDE

The Apache-Netbeans-IDE[2] has a very modular structure and provides many APIs to plugin additional functionality. **Semantic syntax highlighting and braces matching** of the GA DSL is implemented by providing a *TextMate* file so far based on *TextMate Lexer*[16] but an implementation based on the *Language-Server-Protocol*[17] to reach more flexibility is already planned. This will also allow **code completion**, **code snippets**, **error squiggles** and apply **suggestions from errors** just like **general snippets**.

Our GA DSL is based on a *AntLR* grammar[18]. To integrate the *CasADi* expression graphs (created from this at compile time) into the Netbeans Debugger infrastructure is difficult. That´s why we have implemented a second GA implementation based on the same *AntLR* grammar using the *Truffle language implementation framework* [19]. This allows automagically **polyglot debugging** starting from *Java*-code into our DSL with the default Netbeans debugger.

Next steps

It is an ongoing project. The next big steps at the time of writing this abstract are implementing the type system of the DSL and extentions for working with matrices. Furthermore there is a long wishlist specific for GA implementation code generation and optimizations:

- Hyperwedge product implementation following [8] to speed up *CGACasADi*
- Symbolic optimization with *Maxima*[20]
- c-code export and parallelization with *CasADi*
- Using metric matrices to define the ga model instead of cayley-tables
- Completion of CGA implementation based on extended Vahlen matrices following [12]

[15] `https://github.com/MobMonRob/JCasADi` [16] `https://bits.netbeans.org/dev/javadoc/org-netbeans-modules-textmate-l`

[17] `https://en.wikipedia.org/wiki/Language_Server_Protocol`

[18] `https://www.antlr.org/` [19] `https://www.graalvm.org`

[20] `https://maxima.sourceforge.io/`

- ...

References

[1] E. Hitzer and S. Sangwine, *Multivector and multivector matrix inverses in real Clifford algebras.*, In Appl Math Comput., 311, pages 375-389, 2017.

[2] S. De Keninck and M. Roelfs, *Normalization, Square Roots, and the Exponential and Logarithmic Maps in Geometric Algebras of Less than 6D.* In Math Meth Appl Sc., pages 1-17, 2022.

[3] O. Rettig, F. Hinderer and M. Strand, *Application of Conformal Geometric Algebra in Robotics: DH-Parameters Extraction from Joint Axes Poses*, In IAS Society: Proceedings of the 18th International Conference on Intelligent Autonomous Systems IAS-18: Suwon, Korea, pages 4–7, 2023.

[4] E. Hitzer, M. Kamarianakis, G. Papagiannakis and P. Vasik, *Survey of New Applications of Geometric Algebra*, In Math Meth Appl Sc., 2023.

[5] M. Roelfs, S. de Keninck, *Graded Symmetry Groups: Plane and Simple*, In Adv. Appl. Clifford Algebras., 33, 30, 2023.

[6] A. Dargys and A. Acus, *Exponentials of general multivector (MV) in 3D Clifford algebras.* In Nonlinear Anal.: Model. Control., 27, 1, 2022.

[7] L. Dorst, C. Doran, J. Lasenby, *The Inner Products of Geometric Algebra.* In Appl of Geometric Algebra in Comp Sc. and Eng, Chapter, 2002.

[8] S. De Keninck and L. Dorst, *Hyperwedge.* In CGI 2020, LNCS 12221, pages 549-554, 2020.

[9] A. Kleppe and O. Egeland, *Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra*, In Modeling, Identification and Control, 37, 1, pages 63-75, 2016.

[10] I. Zaplana, H. Hadfield and J. Lasenby, *Singularities of serial robots: Identification and distance computation using geometric algebra*, In Mathematics, 10, 2068, 2022.

[11] T. L"ow and S. Calinon, *Geometric Algebra for Optimal Control With Applications in Manipulation Tasks*, In IEEE Transactions on Robotics, 99, pages 1-15, 2022.

[12] L. Dorst *Conformal geometric algebra by extended Vahlen matrices*, In GraVisMa 2009 Workshop proceedings, pages 72-79, 2009

[13] K. Czarnecki and U. W. Eisenecker, *Generative Programming - Methods, Tools, and Applications*, Addison-Wesley Pub Co, 2000.