

# Geometric Algebra Algorithm representing an Orthogonal Isomorphism in versor form v2.03.

Allan Cortzen, c@ctz.dk

## Abstract

From an orthogonal automorphism of a real geometric algebra with indefinite regular metric is constructed an algorithm calculating the corresponding versor. If the algorithm works with an orthonormal basis there are no exceptions.

Some modifications are discussed.

A test for similarities and a versor-like extension to similarities.

Some probability considerations for the definite main case are included.

Mathematics Subject Classification (2000). Primary 15A66; Secondary 15A75.

Keywords. Geometric Algebra, Clifford Algebra, Grassmann Algebra.

## Contents

**Chapter 1. Theoretical considerations**

**Chapter 2. Code. Precision.**

**Conclusion**

**Appendix A. Precision loss in the truncated Algorithm**

**Appendix B. Similarities**

**Appendix C. Conformal geometric Algebra**

**Appendix D. Probability considerations for the main algorithm**

**References**

## Chapter 1 Theoretical considerations

Introduction Preliminaries

The symbolic is the same as in [1],[2], and the reader is assumed familiar with some of the essential results from this material.

Let  $\mathcal{G}(\mathbb{R}^{p,q})$  be the regular geometric algebra over  $\mathbb{R}^{p,q}$ , where  $(p, q)$  is the signature ( $p$  positive) and  $n = p + q$  the dimension of this vector space. We denote reversion by  $R^\sim$  and grade involution by  $R^\wedge$ .

For an orthogonal automorphism  $\psi$  of  $\mathbb{R}^{p,q}$  the algorithm calculates  $V$  in  $\psi(x) = V^\wedge x V^{-1}$ .

From start is given a basis  $\mathcal{A} = (a_i | i = 1 \dots n)$  and its image  $\mathcal{B} = (b_i | i = 1 \dots n)$  by  $\psi$ .

If  $\mathcal{A}$  is orthonormal, then  $a_i^2 = \pm 1$  and  $i \neq j \Rightarrow a_i \cdot a_j = 0$ .

The algorithm is not meant as an actual speed optimized numerical implementation.

## General considerations

The versor  $V$  is constructed from simple reflections and rotations. In calculations we avoid reflections  $a \rightarrow b$ , when  $a$  and  $b$  are nearly identical, and rotations  $a \rightarrow b$ , when  $a$  and  $b$  are nearly opposite. Such transformations may give a serious precision loss.

A reflection along may give loss of relative accuracy for  $a$  close to  $b$  due to the subtraction.

The versor formula for a reflection along  $u$  is  $x \rightarrow -ux / u$ , where  $u \neq 0$  is an invertible vector in  $\mathbb{R}^{p,q}$ .

Assume  $(a - b)$  is invertible. As  $-(a - b)a = b(a - b)$  is verified by expansion, we get after right division  $-(a - b)a / (a - b) = b$ . Therefore a reflection along  $a - b$  maps  $a \rightarrow b$ .

Let  $0 < \delta \leq \sqrt{2}$ . Such a reflection is only used for  $|(a - b)^2| \geq \delta^2$  to avoid too much loss of relative accuracy.

Now assume  $|(a - b)^2| < \delta^2$ . Then a rotation  $a \rightarrow b$  is used constructed from a reflection  $a \rightarrow -b$  followed by a reflection along  $b$ . Smaller  $\delta$  gives fewer of the more calculation costly rotations.

Let  $\kappa = a^2 = b^2 = \pm 1$ . Then  $(a \pm b)^2 = 2(\kappa \pm a \cdot b)$  and  $(a - b)^2 + (a + b)^2 = 4\kappa$ .

As  $\kappa^2 = 1$ ,  $(a - b)^2 \in (-\delta^2, \delta^2) \Rightarrow (a + b)^2 \kappa = 4 - \kappa(a - b)^2 \in (4 - \delta^2, 4 + \delta^2) \Rightarrow |(a + b)^2| > 2$ .

Thus  $a + b$  is a versor without special numeric problems, and the rotation  $a \rightarrow b$  is with done with versor  $b(a + b)$ .

## Norm definition

Let  $(e_i)$  be a *fixed* orthonormal frame. An euclidean metric is defined by  $(e_i)$  being euclidean orthonormal. This gives

a norm such that  $x = \sum \beta_i e_i \Rightarrow \|x\|^2 = \sum \beta_i^2$ , and obviously, as  $\sum \beta_i^2 \geq \sum e_i^2 \beta_i^2$ ,

$$\|x\|^2 \geq |x^2|.$$

Of course, if the metric is definite, then  $\|x\|^2 = |x^2|$ .

## Chapter 2 Code. Precision.

The algorithm shall be proved correct in two cases:

Case 1.  $\mathcal{A}$  is orthonormal.

Case 2. The basis  $\mathcal{A}$  is arbitrary. (The rotation part is not executed.)

The pseudo code for the algorithm

The algorithm do not require  $a_i^2 = b_i^2$  to be normalized.

The number  $\delta \in (0, \sqrt{2}]$  is selected according to a choice between precision and speed.

The algorithm adjusts versor  $V$  in loop  $i$  such that  $a_i \rightarrow b_i$ .

```

V = 1
for i = 1 to n
    R = ai - bi
    d = |R2|
    if d ≥ δ2 |bi2| (* Reflection part *)
        for k = i + 1 to n
            ak = -R ak / R
    else
        if i == n
            break
        if Case_2
            exit("Case 2 failed. Basis permutation could help")

    r = ai + bi (* The rotation part *)
    R = bi r
    for k = i + 1 to n
        ak = R ak / R
V = R V
    
```

The result is a versor  $V$  for  $\psi$ .

As shown later, loosing max.  $s < r$  decimals of significance in  $a_i - b_i$  is secured by choosing  $\delta = \sqrt{n} \cdot 10^{-s}$ .

Proof of correctness,

Case 1.  $\mathcal{A}$  is orthonormal.

Let  $U_i$  be the orthogonal subspace to  $(b_1, \dots, b_{i-1})$ .

First disregard the part “**if**  $i == n$ , **break**”.

We use the following loop invariant of the first **for** loop:

At the start of each iteration  $\Theta(i)$ :  $(a_k = b_k \text{ for } k < i)$  and  $(a_k \in U_i \text{ for } k \geq i)$

*Initialization*: There is nothing to prove, as  $U_1 = \mathbb{R}^{p,q}$ .

*Maintenance*: We assume  $\Theta(i)$  just before the  $i$ th iteration. Thus  $(a_k \in U_i \text{ for } k \geq i)$ .

In this loop  $R$  is constructed from vectors in  $U_i$ , and such that  $a_i \rightarrow b_i$  by  $f_i(x) = \pm R x / R$ . As  $f_i$  fixes vectors orthogonal to  $U_i$ , it also fixes  $b_k = a_k$  for  $k < i$ , and calculation can be avoided.

At the end of the loop  $\Theta(i + 1)$  is satisfied, as  $a_i = b_i$  and, when  $k \geq i + 1$ ,  $a_k \in U_i$  and is orthogonal to  $a_i = b_i$ .

Notice that  $V$  is the total transformation versor of the original  $(a_i)$  frame.

*Termination*: At termination the original  $(a_i)$  frame is transformed with  $V$ , and the result is the  $(b_i)$  frame

The part “**if**  $i == n$ , **break**” is just a simplification:

$\Theta(n)$  implies  $a_n \in U_n$  or  $a_n = \pm b_n$ . If  $a_n = b_n$ , then the the rotation part is superfluous, and  $a_n = -b_n$  is not possible, as the reflection along  $b_n$  has been effectuated, since  $|d| = 4 \geq \delta^2$ .

Case 2. The basis  $\mathcal{A}$  is arbitrary.

To make the proof in Case 1 work here, it is just necessary in loop  $i$  to prove  $R \in U_i$

In this section alone: Let  $\mathcal{A} = (a_{0,i} \mid i = 1 \dots n)$  be the start basis, and fixed, and for any vector  $x$  and any versor  $W$  let  $x^W = W \cdot x \cdot W^{-1}$ .

From orthogonality of  $\psi$  and  $x \rightarrow x^V$  follows  $b_k \cdot b_i = a_{0,k} \cdot a_{0,i} = a_{0,k}^V \cdot a_{0,i}^V$ .

In start of loop  $i$  we have  $a_i = a_{0,i}^V$ . Moreover, for  $k < i$  that  $b_k = a_{0,k}^V$ ,

$b_k \cdot b_i = a_{0,k}^V \cdot a_{0,i}^V = b_k \cdot a_{0,i}^V = b_k \cdot a_i$ . Thus  $b_k \cdot (b_i - a_i) = 0$ , or  $b_k \cdot R = 0$ .

### Possible modifications of the algorithm, if case 2 demands a rotation.

Chose  $\varepsilon > 0$ , and consider a vector  $x$  zero, if  $\|x\| < \varepsilon$ .

Then change the section: **if** Case\_2 ... **exit(...)**

to

```

|R2| ≥ δ2 |bi2|
  if Case_2
    if ||R|| < ε |bi2|
      print("Precision loss")
      continue
    else
      exit("Unstable")

```

NB: If the metric is indefinite it is necessary to use the norm, because  $d < \varepsilon^2$  do not limit the coordinate differences of  $a_i, b_i$ .

In the definite case a simpler possible change is called The truncated Algorithm. Here  $\varepsilon = \delta$ .

```

if Case_2
  print("Precision loss")
  continue

```

### Conclusion

Case 1,  $\mathcal{A}$  orthonormal.

If it is known from start exactly that  $a_i = b_i$  for some indexes these can be omitted.

It is also possible also from a later step, but “exactly” may not be meaningful.

The difference degradation has been brought under control, which is important for general applications.

Case 2,  $\mathcal{A}$  an arbitrary basis.

Even if  $\delta$  is optimized roughly half of the precision is lost in typical cases.

The non-definite case may be unstable.

## Appendix A. Precision loss

Assume the unit vectors  $a, b$  are represented with uncertainty  $\xi = 0.5 \cdot 10^{-h}$ , that is  $\|\Delta a\| = \|\Delta b\| = \xi$ . Then the relative uncertainty is approximately the same. From this follows  $\|\Delta(a - b)\| \leq 2\xi$ .

In the reflection section  $\|a - b\|^2 \geq (a - b)^2 \geq \delta^2$ . Thus the relative uncertainty for  $a - b$  in the worst situation is  $2\xi/\delta$ .

If the relative angle error of  $V$  from each of  $n - 1$  loops is  $\tau$ , then the total angle error is very roughly less than  $\tau n$ .

Case 1. The main source here is difference degradation, i.e. loss of significance, by the subtraction  $a - b$ .

Loosing max.  $s < h$  decimals of significance is secured by choosing  $\delta = 2 \cdot 10^{-s}$  in the algorithm, as  $2\xi/\delta \leq 10^s \xi \Leftrightarrow \delta \geq 2 \cdot 10^{-s}$ .

Case 2. The truncated Algorithm.

Precision degradation here are mainly two folds and also present in [3] algorithm 1 and 2.

Assume that  $\varepsilon = \delta = 10^{-k}$ ,  $0 < k \leq h$ .

1. (Difference degradation) Assume  $\|a - b\| = 1.01 \delta$  then the reflection is executed.

The relative error from reflection in the worst situation is  $2\xi/(1.01 \delta) \simeq 10^{-h}/10^{-k} = 10^{k-h}$ , and a angle error of roughly this size affect  $V$  in the worst case.

2. (Non-equality degradation) Assume  $\|a - b\| = 0.99 \delta \simeq 10^{-k}$ , then equality of  $a$  and  $b$  is assumed, and a angle error of roughly this size affect  $V$  in the worst case.

3. The balance point for this dilemma is  $k \simeq h/2$ .

## Appendix B. Similarities [4]

Similarities are essential for image processing.

Vector space similarities created by composing an orthogonal automorphism and a multiplication with some factor  $\lambda > 0$ .

From start is given a basis  $\mathcal{A} = (a_i | i = 1 \dots n)$  and its image  $\mathcal{B} = (b_i | i = 1 \dots n)$  by a *similarity*  $\phi$ .

Let  $\psi$  be the corresponding orthogonal automorphism.

Then  $\phi = \lambda \psi$ , and  $a_i \cdot a_j = \lambda^{-2} b_i \cdot b_j$  for  $i \leq j$ , and both in  $\{1 \dots n\}$ .

Therefore

$$\lambda = \sqrt{(\sum_{i \leq j} |b_i \cdot b_j|) / (\sum_{i \leq j} |a_i \cdot a_j|)}.$$

A simple extension of the code to similarities is to prepend the algorithm with this formula and the lines

```
for  $i = 1$  to  $n$ 
   $b_i = b_i / \lambda$ 
```

The resulting formula then becomes  $\phi(x) = \lambda V^x V^{-1}$

The algorithm only require  $a_i^2$  to be normalized within a common factor, like the main algorithm.

### Appendix C. Conformal geometric Algebra [4]

The conformal geometric algebra representing  $m$ -dimensional euclidean space,  $\mathbb{R}^m$ , is  $\mathcal{G}(\mathbb{R}^{m+1,1})$ .

Assume  $m \geq 2$ . The conformal transformations of  $\mathbb{R}^m$  can be represented by orthogonal isomorphisms of  $\mathbb{R}^{m+1,1}$ .

Let  $n = m + 2$ .

From start is given a basis  $\mathcal{A} = (a_i | i = 1 \dots n)$  and its image  $\mathcal{B} = (b_i | i = 1 \dots n)$  by a *conformal point transformation*  $\chi$ .

Let  $\psi$  be the corresponding orthogonal automorphism.

As each point in  $\mathbb{R}^m$  corresponds to a 1-dimensional subspace in  $\mathbb{R}^{m+1,1}$  the parameters  $(\lambda_i | i = 1 \dots, n)$  in  $\psi(a_i) = \lambda_i b_i$  must be determined before the algorithm can be used.

For null vectors  $x, y$  holds  $x \cdot y = (x - y)^2$ . By proper injection of  $\mathbb{R}^m$  in  $\mathcal{G}(\mathbb{R}^{m+1,1})$  we may assume  $\mathcal{A}$  don't contain  $\infty$ .

Thus  $a_i \cdot a_j = \lambda_i \lambda_j b_i \cdot b_j \neq 0$ . Now we can define  $k_{i,j} = \lambda_i \lambda_j = b_i \cdot b_j / (a_i \cdot a_j)$ , calculate  $\lambda_1$  from  $k_{1,2} k_{1,3} / k_{2,3} = \lambda_1^2$ , and for  $i > 1$  find  $\lambda_i$  from  $k_{1,i} = \lambda_1 \lambda_i$ .

This gives  $\lambda_1 = \sqrt{k_{1,2} k_{1,3} / k_{2,3}}$  and  $(\lambda_2, \dots, \lambda_n) = \pm(k_{1,2}, \dots, k_{1,n}) / \lambda_1$

A simple extension of the code to conformal transformations is to prepend the algorithm with these calculations and the lines

```
for  $i = 1$  to  $n$ 
   $b_i = \lambda_i a_i$ 
```

The relevant formula then becomes  $\psi(x) = V^x V^{-1}$

The algorithm only require  $a_i^2$  to be normalized within a common factor, like the main algorithm.

If  $\psi$  is a rotation we obviously can reduce  $n$  by 1.

### Appendix D. Probability considerations for the main algorithm

The purpose is to calculate the probability for use of the rotational code of course dependent of  $\delta$  and certain conditions e.g. uniformity of distribution. It will show that rotations are strongly decreasing with loops are remaining and with decreasing  $\delta$ .

Binary truncation is close to decimal rounding.

In the definite case loosing decimals by subtraction  $R = a_i - b_i$  are given in the table and found from the formula last in this paragraph:

decimals	$\delta$	Probability for rotation choice	
max ½ decimal	$\delta = 0.632$	$(0.10, 0.025, 0.0067)$	when (2,3,4) loops are remaining
max 1 decimal	$\delta = 0.2$	$(0.032, 0.0025, 0.00021)$	when (2,3,4) loops are remaining
max 1½ decimal	$\delta = 0.063$	$(0.010, 0.00025, 6.7 \times 10^{-6})$	when (2,3,4) loops are remaining

Thus the occurrence of a rotation is mainly in the second to last loop and here with probability 3.2%, if loosing max 1 decimal.

Derivation of the result.

We assume here  $(b_i)$  is the standard basis for  $\mathbb{R}^{p,q}$ , and use concepts  $B_i$  and  $f_i$  from the proof section.

It may be of interest to find the rotation probability in loop  $i$  relative to the left-invariant Haar measure  $\mu$  on the orthogonal group  $\mathbb{G} = O(\mathbb{R}^{p,q})$ ,  $n = p + q \geq 2$ . Thus, if  $g \in \mathbb{G}$ , and  $\mathbb{A}$  is a borel subset of  $\mathbb{G}$ , then  $\mu(g \mathbb{A}) = \mu(\mathbb{A})$ .

On elements in the product of unit surfaces  $S_m = \{x \in \mathbb{R}^{p,q} \mid |x^2| = 1\}^{\times k}$ , denoted  $\underline{x} = (x_1, \dots, x_m)$ , we have a natural left group action,  $(g, \underline{x}) \rightarrow g \underline{x} = (g x_1, \dots, g x_m)$ ,  $g \in \mathbb{G}$

The image of  $\mu$  by the continuous mapping  $f_0 : h \rightarrow h \underline{a}$ ,  $h \in \mathbb{G}$  gives a measure  $\mu_0$  on  $S_n$ .

If  $g \in \mathbb{G}$  and  $\mathbb{B}$  is a borel subset of  $S_n$ , then we have  $\mu_0(g \mathbb{B}) = \mu(f_0^{-1}(g \mathbb{B})) = \mu(g f_0^{-1}(\mathbb{B})) = \mu(f_0^{-1}(\mathbb{B})) = \mu_0(\mathbb{B})$ , which shows

$\mu_0$  is  $\mathbb{G}$  left invariant.

Let  $\mathbb{G}_i$  be the subgroup of  $\mathbb{G}$  fixing  $b_1, \dots, b_{i-1}$  and  $S^i = \{(b_1, \dots, b_i)\} \times S_{n-i}$  and  $\mathbb{B}_i$  is a borel subset of  $S^i$ .

The transformation  $f_1$ , acting on  $S_n$ , maps the measure  $\mu_0$  onto a measure  $\mu_1$  on  $S^1 = \{b_1\} \times S_{n-1}$ , which is  $\mathbb{G}_1$  left invariant, as

$$\mu_1(g_1 \mathbb{B}_1) = \mu_0(f_1^{-1}(g_1 \mathbb{B}_1)) = \mu_0(g_1 f_1^{-1}(\mathbb{B}_1)) = \mu_0(f_1^{-1}(\mathbb{B}_1)) = \mu_1(\mathbb{B}_1).$$

By induction follows easily:

The transformation  $f_i$ , acting on  $S_n$ , maps the measure  $\mu_{i-1}$  onto a measure  $\mu_i$  on  $S^i$ , which is  $\mathbb{G}_i$  left invariant, as

$$\mu_i(g_i \mathbb{B}_i) = \mu_{i-1}(f_i^{-1}(g_i \mathbb{B}_i)) = \mu_{i-1}(g_i f_i^{-1}(\mathbb{B}_i)) = \mu_{i-1}(f_i^{-1}(\mathbb{B}_i)) = \mu_i(\mathbb{B}_i).$$

Precisely if the metric on  $B_i$  is definite,  $\mu_i$  can be normalized to a probability measure on  $S^i$  or obviously on  $S_{n-i}$  by a bijective projection.

Considerations for  $S_m = S_{n-i}$  under definite metrics on  $B_i$ .

The unit sphere  $\{x \mid |x^2| = 1\}$  is parametrized with angular coordinates  $\varphi_1, \dots, \varphi_{m-2}, |\varphi_{m-1}| \in [0, \pi]$  as

$x_1 = \cos(\varphi_1), \dots, x_{m-1} = \sin(\varphi_1) \cdots \sin(\varphi_{m-2}) \cos(\varphi_{m-1}), x_m = \sin(\varphi_1) \cdots \sin(\varphi_{m-2}) \sin(\varphi_{m-1})$  with metric density  $g \approx |\sin^{m-2}(\varphi_1) \dots \sin^2(\varphi_{m-3}) \sin(\varphi_{m-2})|$ .

The marginal density for  $\varphi_1$  is

$$F_m(\varphi) = \frac{1}{F_m} \int_0^\varphi \sin^{m-2}(\varphi_1) d\varphi_1,$$

where  $F_m = \int_0^\pi \sin^{m-2}(\varphi_1) d\varphi_1 = \sqrt{\pi} \Gamma(\frac{m-1}{2}) / \Gamma(\frac{m}{2})$ ,

valid also for  $|\varphi_1|$ , if  $m = 2$ . The angle to be used in  $F$  is  $\varphi = \arcsin \sqrt{\delta^2 - \delta^{2^2} / 4}$ .

## References

- [1] Allan Cortzen. Direct Construction of Clifford and Geometric Algebras and their basic algebraic Structures. 2016.  
<http://ctz.dk/geometric-algebra/clifford-geometric-algebras-algebraic-structures/>
- [2] L. Dorst, D. Fontijne, and S. Mann. Geometric Algebra for Computer Science. An Object-Oriented Approach to Geometry. Morgan Kaufman, 2007.
- [3] Leo Dorst and Joan Lasenby, editors. Guide to Geometric Algebra in Practice , p. 63-78. Springer London, 2011.
- [4] Alexander Arsenovic personal communications. 2016.