Geometric Algebra Computing

Der Weg zu Gaalop

17.07.2019

Dr. Dietmar Hildenbrand

Technische Universität Darmstadt

Implementierung von geometrischer Algebra

TECHNISCHE UNIVERSITÄT DARMSTADT

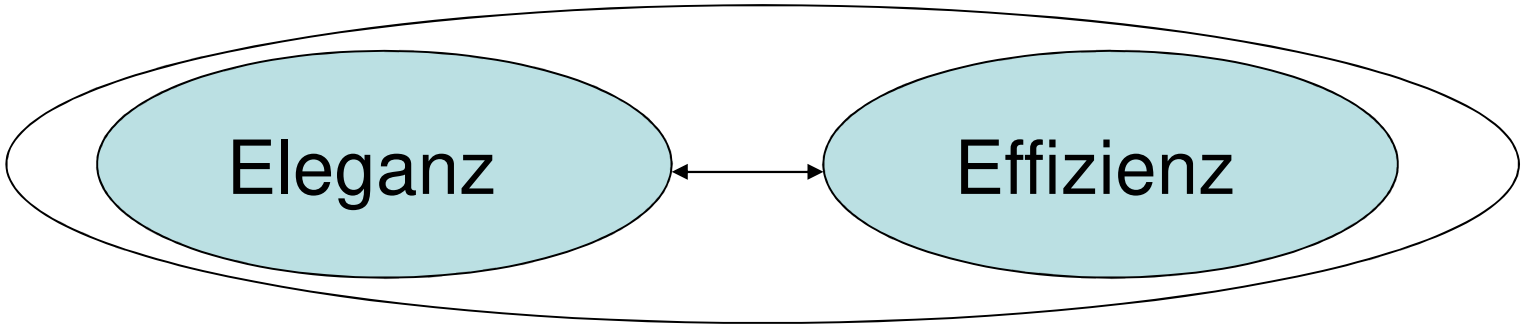
- **Stand 2003**
  - Elegant, aber langsam?
    (**Gaigen**-Bsp. mit Raytracer) [Fontijne, Dorst 2003]

model	implementation	full rendering time
3D LA	standard	$1.00\times(23.3\text{s})$
3D GA	Gaigen	$2.56\times$
4D LA	standard	$1.05\times$
4D GA	Gaigen	$2.97\times$
5D GA	Gaigen	$5.71\times$

  - Hardware nicht auf geometrische Algebra ausgelegt
  - 2^n Basiselemente (32 für konforme geometrische Algebra)
    (Bsp. 3D-Vektor → 32D-Vektor)

Index	blade
1	1
2	e_1
3	e_2
4	e_3
5	e_∞
6	e_0
7	$e_1 \wedge e_2$
8	$e_1 \wedge e_3$
9	$e_1 \wedge e_\infty$
10	$e_1 \wedge e_0$
11	$e_2 \wedge e_3$
12	$e_2 \wedge e_\infty$
13	$e_2 \wedge e_0$
14	$e_3 \wedge e_\infty$
15	$e_3 \wedge e_0$
16	$e_\infty \wedge e_0$
17	$e_1 \wedge e_2 \wedge e_3$
18	$e_1 \wedge e_2 \wedge e_\infty$
19	$e_1 \wedge e_2 \wedge e_0$
20	$e_1 \wedge e_3 \wedge e_\infty$
21	$e_1 \wedge e_3 \wedge e_0$
22	$e_1 \wedge e_\infty \wedge e_0$
23	$e_2 \wedge e_3 \wedge e_\infty$
24	$e_2 \wedge e_3 \wedge e_0$
25	$e_2 \wedge e_\infty \wedge e_0$
26	$e_3 \wedge e_\infty \wedge e_0$
27	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty$
28	$e_1 \wedge e_2 \wedge e_3 \wedge e_0$
29	$e_1 \wedge e_2 \wedge e_\infty \wedge e_0$
30	$e_1 \wedge e_3 \wedge e_\infty \wedge e_0$
31	$e_2 \wedge e_3 \wedge e_\infty \wedge e_0$
32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$

Läßt sich Eleganz mit Effizienz vereinbaren?



Wo stehen wir 2005?

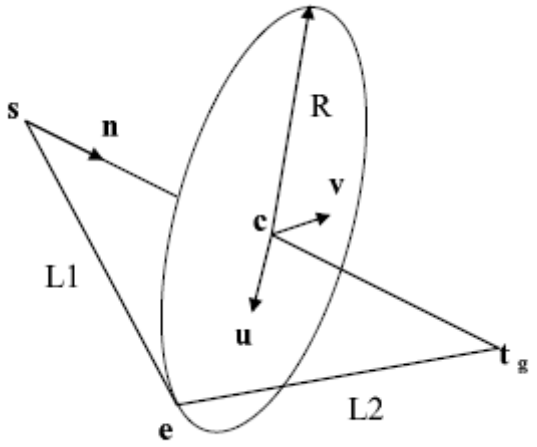
TECHNISCHE UNIVERSITÄT DARMSTADT

Untersuchte Anwendung

- Projekt Virtual Human
- Inverse Kinematik des Arms

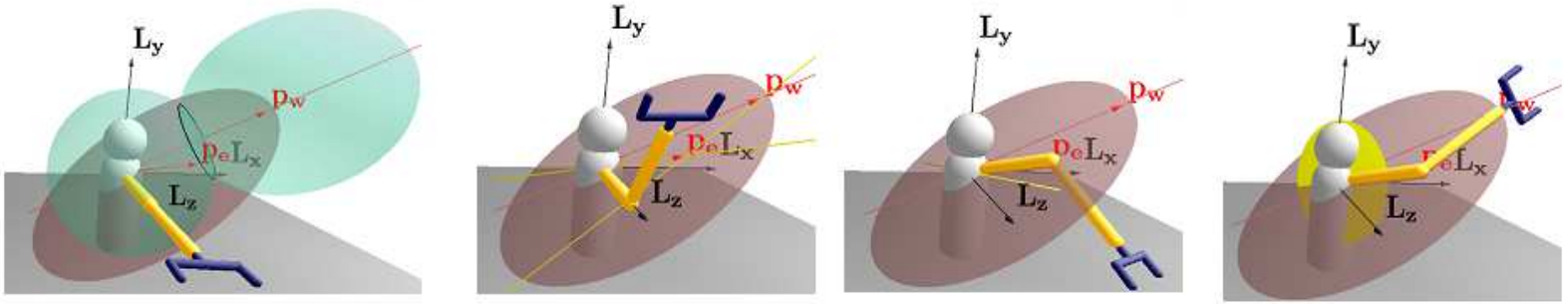
[Virtual Human]

- IKAN-Bibliothek
  - Analytischer, geometrischer Algorithmus
  - [Tolani et al. 2000]



Wo stehen wir 2005?

TECHNISCHE UNIVERSITÄT DARMSTADT



- Interaktive, visuelle Entwicklung
- Kompakte Algorithmen (ca. 20-30%)
- Schneller Prototyp und Test
- [Hildenbrand et al. ICRA 2005]
- Erste Implementierung mit Hilfe von Gaigen
  [Fontijne, Dorst 2002]
- Zunächst 50 mal langsamer als die herkömmliche Lösung !

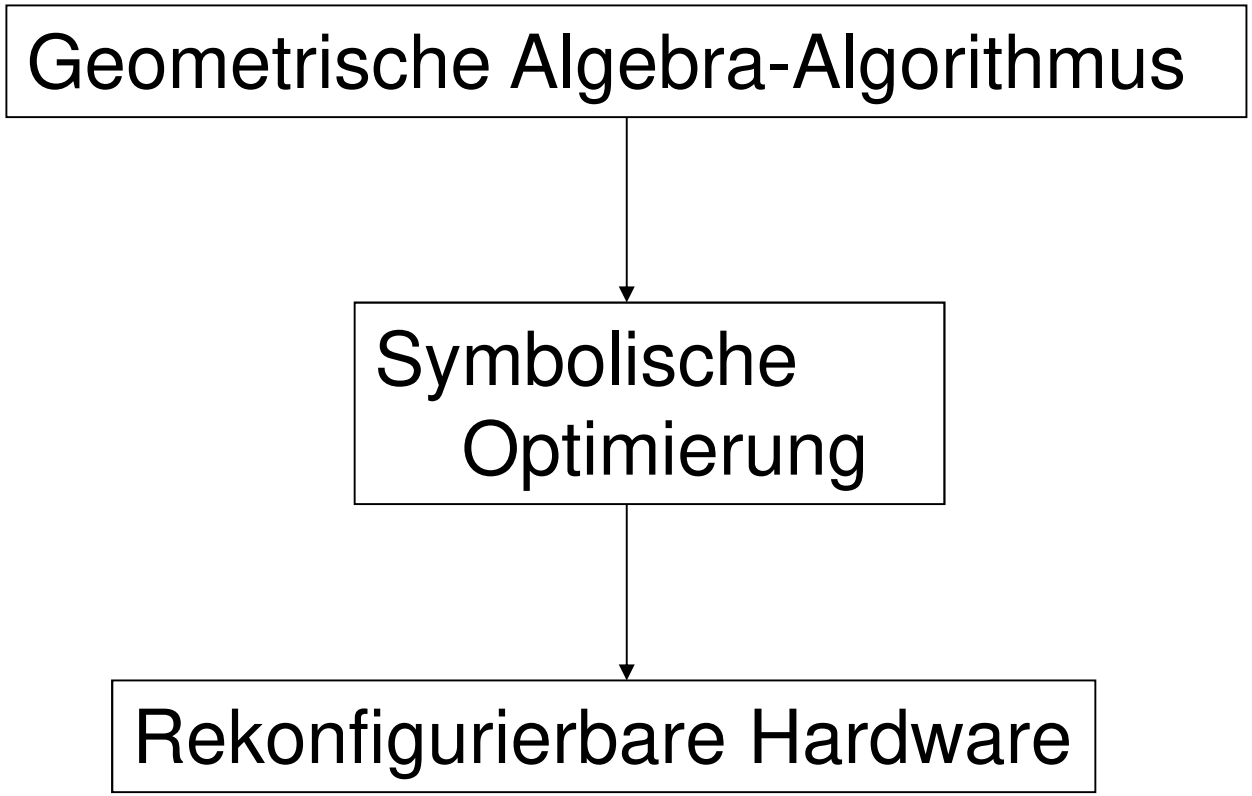
Wege zur Effizienz

TECHNISCHE UNIVERSITÄT DARMSTADT

- Algorithmus-Ebene
  - Einbettung von Quaternionen
  - Wenige, kompakte Formulierungen
  - Vermeidung trigonometrischer Funktionen
  - [Hildenbrand et al. ICCA 2005]

Wege zur Effizienz

TECHNISCHE UNIVERSITÄT DARMSTADT



C-Code

FPGA

Implementierungen von C-Code und Verilog-Code zunächst von Hand!

Optimierung mit Hilfe von Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Computer Algebra System
- Erlaubt symbolisches Rechnen

```
> func:=(a^3+3*a^2*b+3*a*b^2+b^3)/(a^2+2*a*b+b^2);
```

$$func := \frac{a^3 + 3a^2b + 3ab^2 + b^3}{a^2 + 2ab + b^2}$$

```
> simplify(func);
```

$$a + b$$

```
>
```

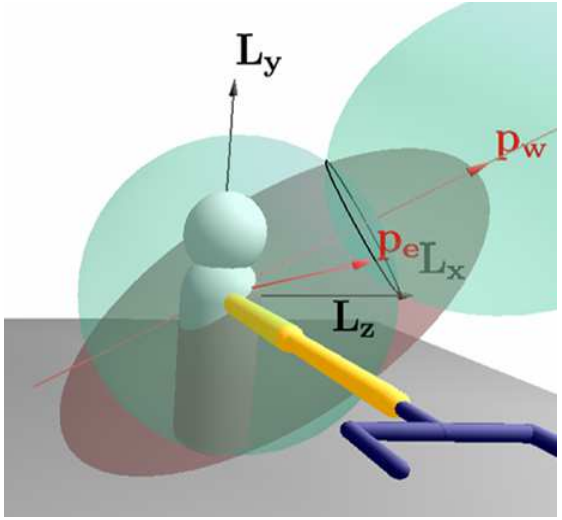
- Cliffordlib für Geometrische Algebra (Fauser/Ablamowicz)
- Ziel: Reduktion von Ausdrücken bis zu den einfachsten Additionen und Multiplikationen

Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
        (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```

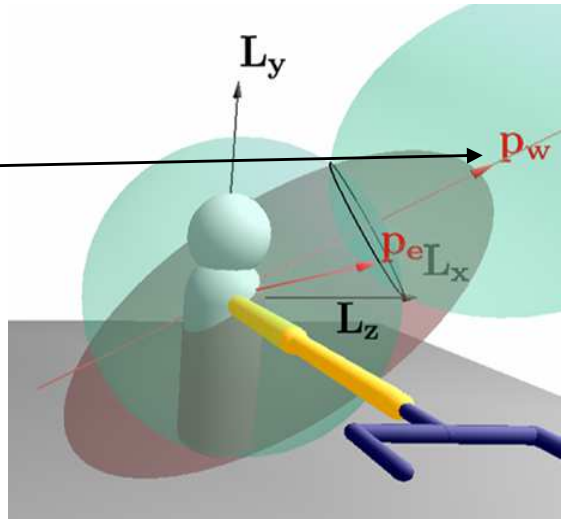


Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
        (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```

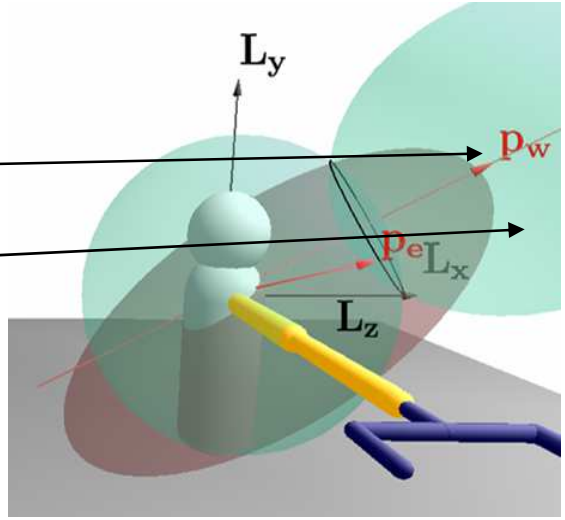


Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
      (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```

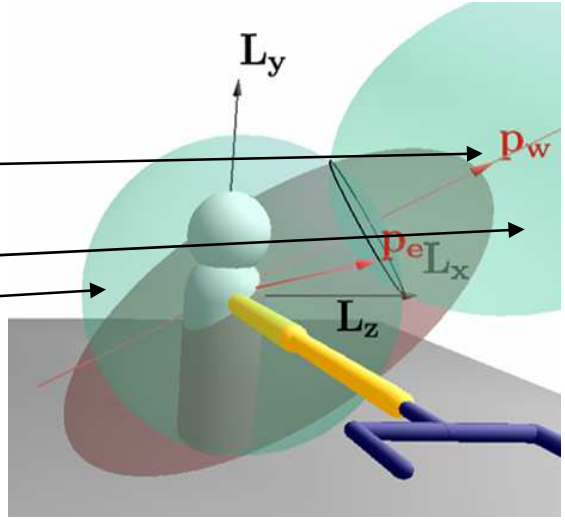


Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
        (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```

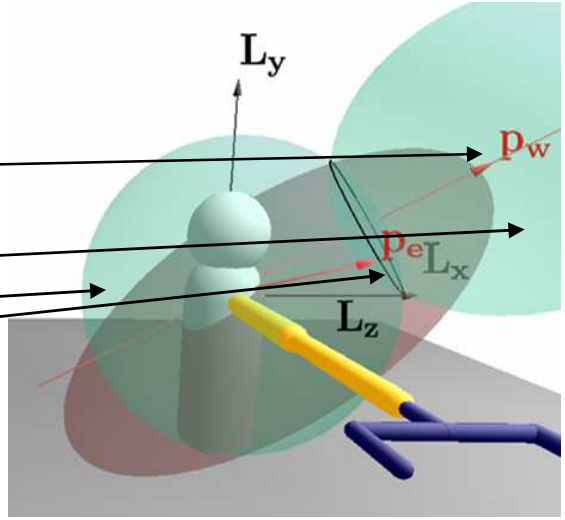


Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
        (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```

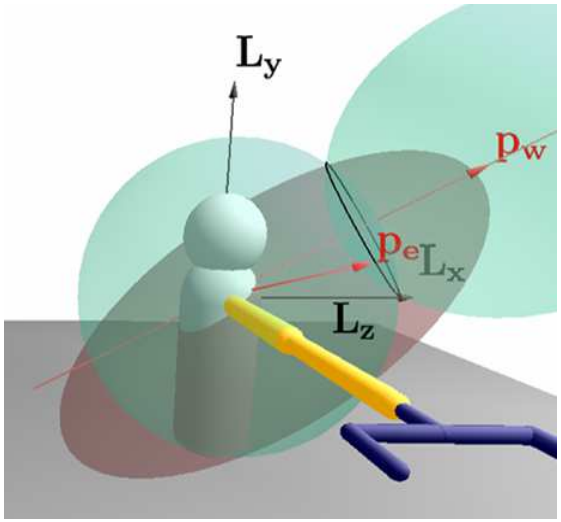


Beispiel für symbolische Vereinfachung mit Maple

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute the elbow point p_e

```
> pw:=pwx*e1+pwy*e2+pwz*e3+0.5*
        (pwx^2+pwy^2+pwz^2)*einf+e0;
> S1:=pw-0.5*L2*L2*einf;
> S2:=e0-0.5*L1*L1*einf;
> C_e:=S1 &w S2;
```



Simplify(C_e)

C_e[9] := -1/2*pwx*L1^2 C_e[10] := pwx … C_e[16] := …

Index	blade	Index	blade	Index	blade	Index	blade	Index	blade	.....	.....	Index	blade
1	1	2	e_1	3	e_2	4	e_3	5	e_∞	..	..	32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$

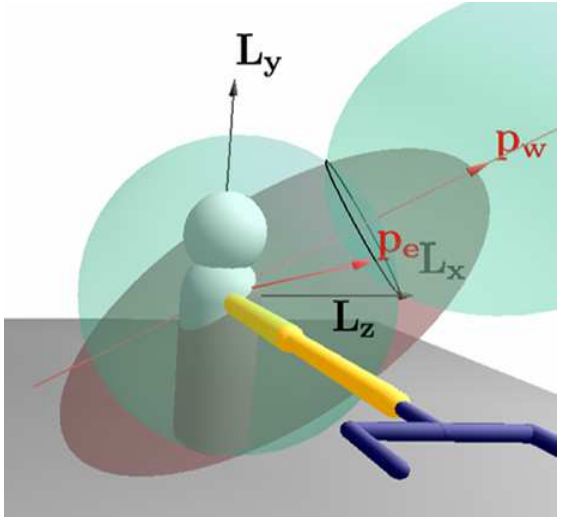
Beispiel für symbolische Vereinfachung mit Maple …

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ellbogenkreis bereits berechnet …
- Schnitt mit Swivel Plane
- Resultat: Punktpaar
- Auswahl eines Punktes führt zu:

$$
\begin{aligned}
p_{ex} &= (PP_j(PP_{34} - PP_{35}) + PP_k(PP_{25} - PP_{24} \\
&\quad + tmp_{sqrt}(PP_{15} - PP_{14}))/einf\_PP \\
p_{ey} &= (PP_i(PP_{35} - PP_{34}) + PP_k(PP_{14} - PP_{15}) \\
&\quad + tmp_{sqrt}(PP_{25} - PP_{24}))/einf\_PP \\
p_{ez} &= (PP_j(PP_{15} - PP_{14}) + PP_i(PP_{24} - PP_{25}) \\
&\quad + tmp_{sqrt}(PP_{35} - PP_{34}))/einf\_PP
\end{aligned}
$$

(siehe späteres Hardware-Beispiel)



Eurographics-Paper 2006, Wien

TECHNISCHE UNIVERSITÄT DARMSTADT

- **" Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra "**
- Dietmar Hildenbrand, Daniel Fontijne, Yusheng Wang, Marc Alexa and Leo Dorst
- Erster Nachweis, dass ein Algorithmus in GA schneller sein kann als ein herkömmlicher Algorithmus

Hardware-Implementierung

TECHNISCHE UNIVERSITÄT DARMSTADT

Index	blade
1	1
2	e_1
3	e_2
4	e_3
5	e_∞
6	e_0
7	$e_1 \wedge e_2$
8	$e_1 \wedge e_3$
9	$e_1 \wedge e_\infty$
10	$e_1 \wedge e_0$
11	$e_2 \wedge e_3$
12	$e_2 \wedge e_\infty$
13	$e_2 \wedge e_0$
14	$e_3 \wedge e_\infty$
15	$e_3 \wedge e_0$
16	$e_\infty \wedge e_0$
17	$e_1 \wedge e_2 \wedge e_3$
18	$e_1 \wedge e_2 \wedge e_\infty$
19	$e_1 \wedge e_2 \wedge e_0$
20	$e_1 \wedge e_3 \wedge e_\infty$
21	$e_1 \wedge e_3 \wedge e_0$
22	$e_1 \wedge e_\infty \wedge e_0$
23	$e_2 \wedge e_3 \wedge e_\infty$
24	$e_2 \wedge e_3 \wedge e_0$
25	$e_2 \wedge e_\infty \wedge e_0$
26	$e_3 \wedge e_\infty \wedge e_0$
27	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty$
28	$e_1 \wedge e_2 \wedge e_3 \wedge e_0$
29	$e_1 \wedge e_2 \wedge e_\infty \wedge e_0$
30	$e_1 \wedge e_3 \wedge e_\infty \wedge e_0$
31	$e_2 \wedge e_3 \wedge e_\infty \wedge e_0$
32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$

- Ausnutzung der Parallelität von Hardware !

Index	blade	Index	blade	Index	blade	Index	blade	Index	blade	.....	.....	Index	blade
1	1	2	e_1	3	e_2	4	e_3	5	e_∞	..	..	32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$

HW-Implementierung

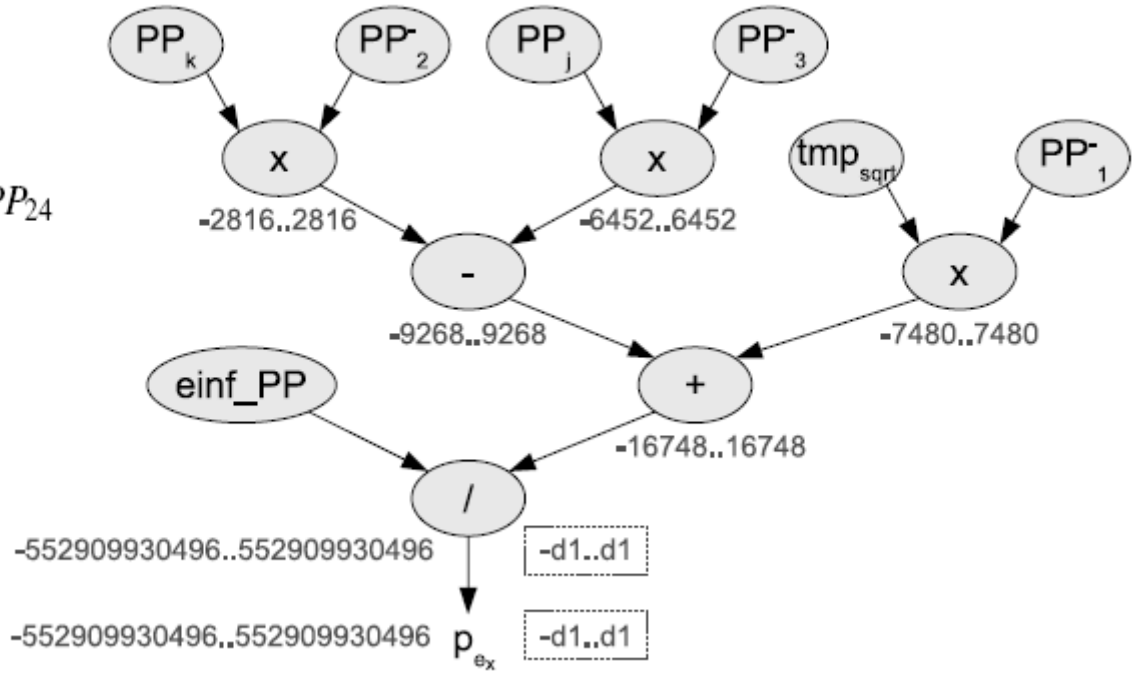
TECHNISCHE UNIVERSITÄT DARMSTADT

- Datenflußgraph
  - für

$$p_{ex} = (PP_j(PP_{34} - PP_{35}) + PP_k(PP_{25} - PP_{24} + tmp_{sqrt}(PP_{15} - PP_{14}))/einf\_PP$$

  - mit Vereinfachungen wie

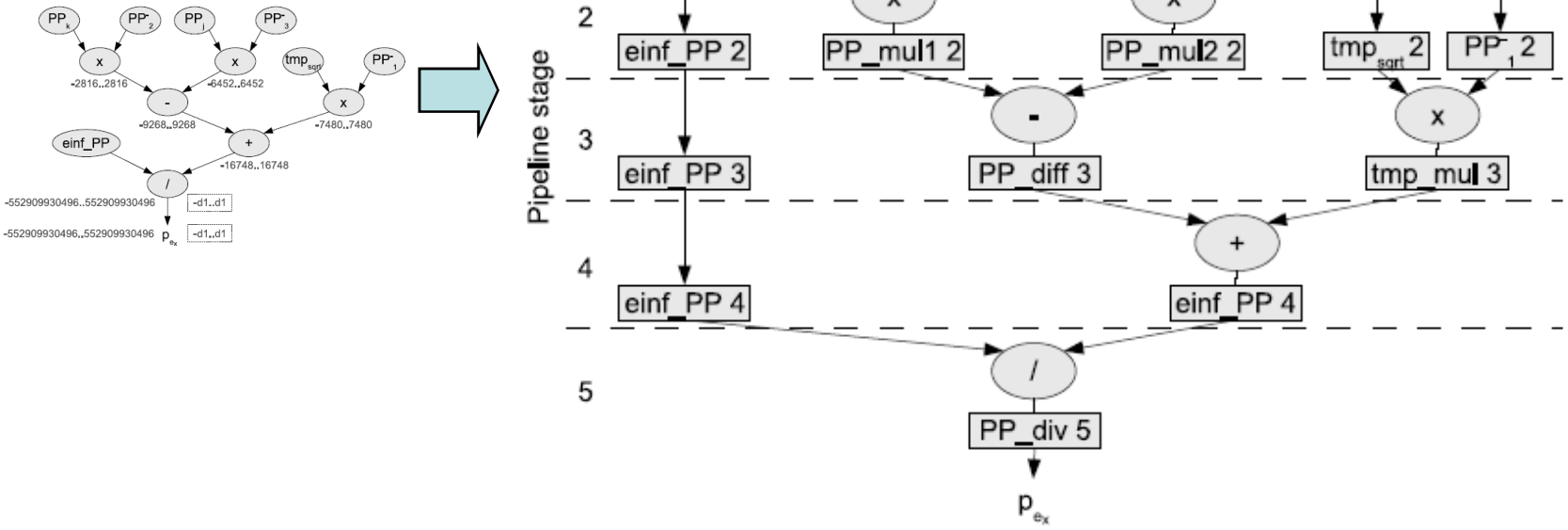
$$PP_m^- = PP_{m5} - PP_{m4}$$



  - Gültigkeits-Bereiche der Werte bei FPGA wichtig!

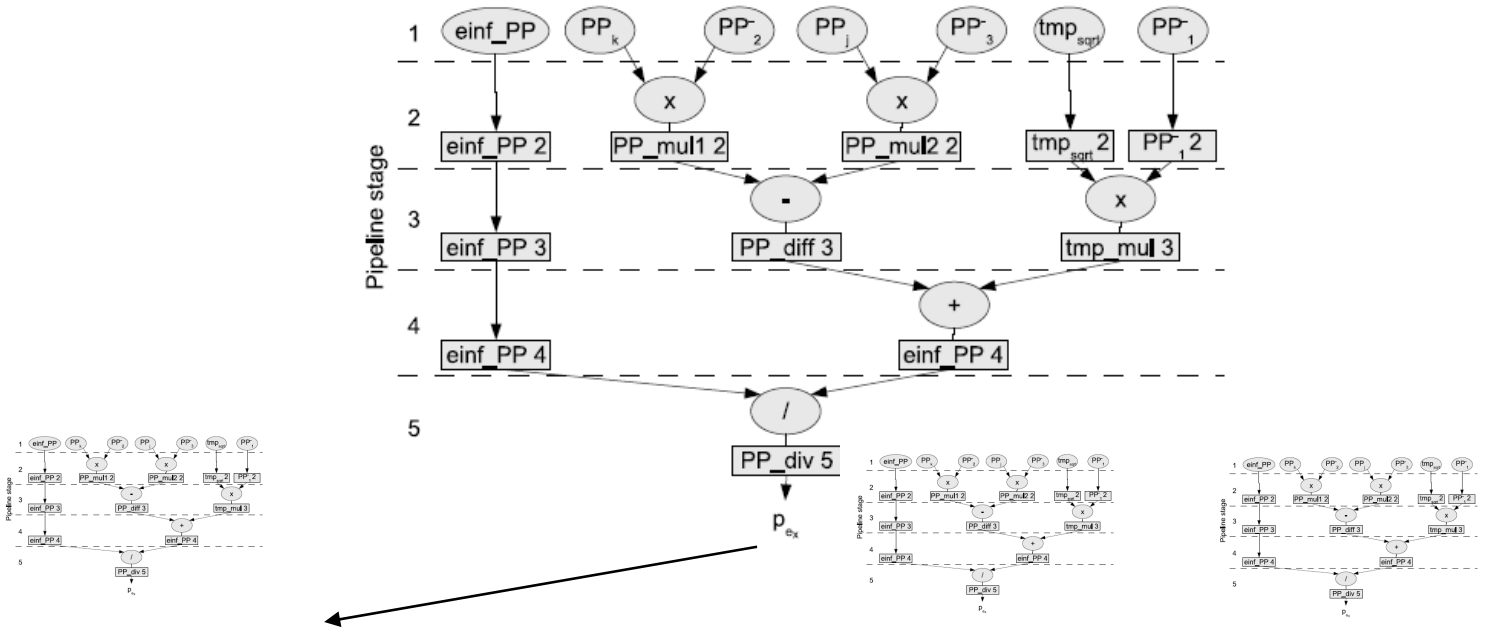
HW-Implementierung

- Pipeline-Stufen



Parallelität in 2 Dimensionen

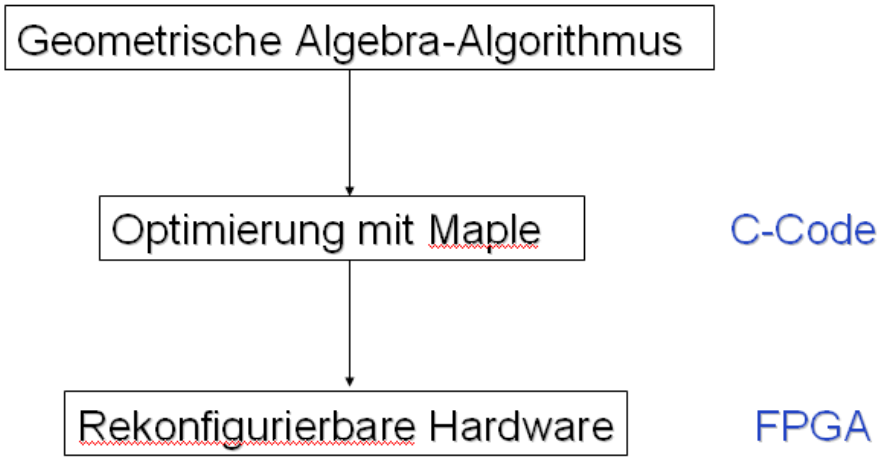
- 363 Pipeline-Stufen, bis zu 20 skalare Operationen pro Stufe



Index	blade	Index	blade	Index	blade	Index	blade	Index	blade	.....	.....	Index	blade
1	1	2	e_1	3	e_2	4	e_3	5	e_∞	..	..	32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$

Stand 2008

TECHNISCHE UNIVERSITÄT DARMSTADT



- Maple-optimierte Implementierung ca. *drei mal* schneller als die herkömmliche Lösung [Hildenbrand et al. Eurographics 2006]
- Rekonfigurierbare Hardware: nochmals um den *Faktor 100* schneller [Hildenbrand et al. Grapp 2008]
- -> Anwendung insgesamt um den *Faktor 300* schneller als die herkömmliche Lösung! (*x15000* zur ersten Implementierung!)

Gaalop 2010

TECHNISCHE UNIVERSITÄT DARMSTADT

Geometric **a**lgebra **al**gorithms ***op***timizer (www.Gaalop.de)

- Ziel:
  - Kombination von einfachem geometrischem Rechnen mit hoher Performanz der Implementierung

- Dietmar Hildenbrand, Joachim Pitt and Andreas Koch,
  Kapitel **"Gaalop - High Performance Parallel Computing based on Conformal Geometric Algebra"**
  in **Geometric Algebra Computing for Engineering and Computer Science**, Springer Verlag 2010, E. Bayro-Corrochano and G. Scheuermann.

Gaalop

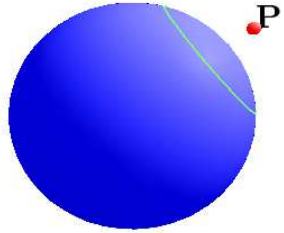
TECHNISCHE UNIVERSITÄT DARMSTADT

- interpretiert geometrische Algebra - Inputfile
- generiert optimierte Implementierung
- Kommandozeilen-Version für Gaalop GPC

Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Berechnung des Horizontkreises:
  - point P = VecN3(x,y,z)
  - sphere $S = e_0 - \frac{1}{2}r^2 e_\infty$
  - $C = S \wedge (P + (P \cdot S)e_\infty)$



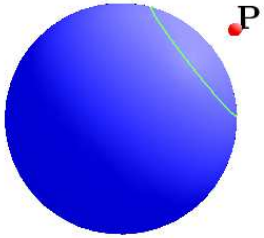
Gaalop-Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Berechnung des Horizontkreises:
  - point P = VecN3(x,y,z)
  - sphere $S = e_0 - \frac{1}{2}r^2 e_\infty$
  - $C = S \wedge (P + (P \cdot S)e_\infty)$
  - or
  - $C = S \wedge (S \cdot (P \wedge e_\infty))$ [Leo Dorst]



Horizont-Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

$$C = S \wedge (P + (P \cdot S)e_\infty)$$

$$C = S \wedge (S \cdot (P \wedge e_\infty)) \text{ [Leo Dorst]}$$

Horizont-Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

$$C = S \wedge (P + (P \cdot S)e_\infty)$$

```
P =  VecN3(x,y,z);
S = e0-0.5*r*r*einf;
?C=S^(P+(P.S)*einf);
```

$$C = S \wedge (S \cdot (P \wedge e_\infty))$$ [Leo Dorst]

```
P =  VecN3(x,y,z);
S = e0-0.5*r*r*einf;
?C =S^(S.(P^einf));
```

Horizont-Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

$C = S \wedge (P + (P \cdot S)e_\infty)$

```
P =  VecN3(x,y,z);
S = e0-0.5*r*r*einf;
?C=S^(P+(P.S)*einf);
```

$C = S \wedge (S \cdot (P \wedge e_\infty))$ [Leo Dorst]

```
P =  VecN3(x,y,z);
S = e0-0.5*r*r*einf;
?C =S^(S.(P^einf));
```

```
float C_opt[32] = {0.0};
C_opt[9]=0.5*x*r*r;
C_opt[10]=-1*x;
C_opt[12]=0.5*y*r*r;
C_opt[13]=-1*y;
C_opt[14]=0.5*z*r*r;
C_opt[15]=-1*z;
C_opt[16]=-1*r*r;
```

- Anm.: verschiedene Formeln -> gleiches optimiertes Ergebnis
- Anm.: Gaalop optimiert ganze Teile von Algorithmen!

Gaalop 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

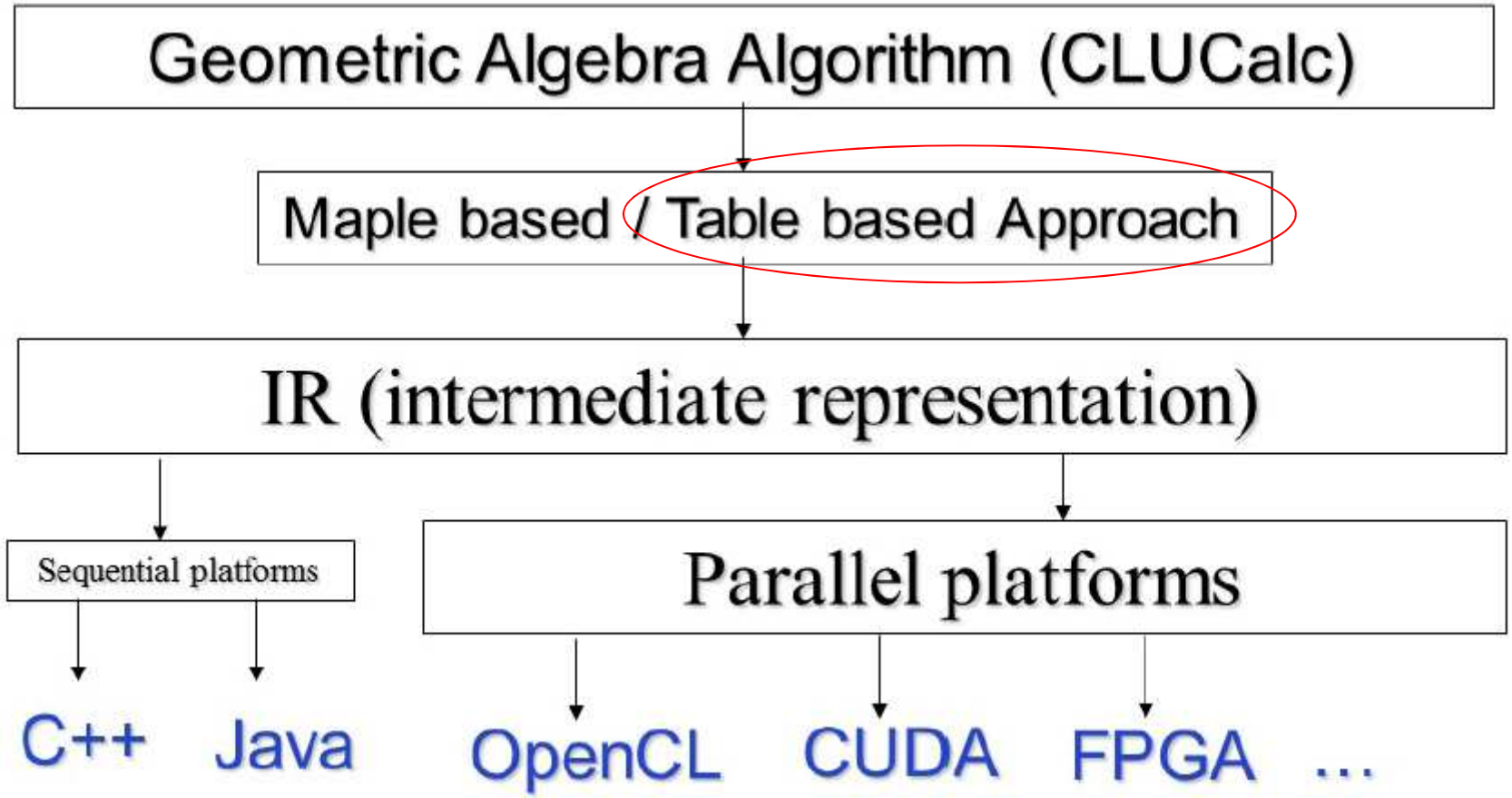


Table based Compilation Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- Geometric product multiplication table in 3D GA:

		b		$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$				
			E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
a			1	e_1	e_2	e_3	e_{12}	e_{23}	e_{13}	e_{123}
	E_1	1	0	0	0	0	0	0	0	0
$\mathbf{a_1}$	E_2	e_1	0	E_1	E_5	E_7	0	0	0	0
$\mathbf{a_2}$	E_3	e_2	0	$-E_5$	E_1	E_6	0	0	0	0
$\mathbf{a_3}$	E_4	e_3	0	$-E_7$	$-E_6$	E_1	0	0	0	0
	E_5	e_{12}	0	0	0	0	0	0	0	0
	E_6	e_{23}	0	0	0	0	0	0	0	0
	E_7	e_{13}	0	0	0	0	0	0	0	0
	E_8	e_{123}	0	0	0	0	0	0	0	0

- GA algorithm:

```
a=a1*e1+a2*e2+a3*e3;
b=b1*e1+b2*e2+b3*e3;
?c=a*b;
```

resulting C code:

```
c[1]=a1*b1+a2*b2+a3*b3;
c[5]=a1*b2-a2*b1;
c[6]=a2*b3-a3*b2;
c[7]=a1*b3-a3*b1;
```

Table based Compilation Example

TECHNISCHE UNIVERSITÄT DARMSTADT

- Geometric product multiplication table in 3D GA:

		b		$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_3}$				
			E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
a			1	e_1	e_2	e_3	e_{12}	e_{23}	e_{13}	e_{123}
	E_1	1	0	0	0	0	0	0	0	0
$\mathbf{a_1}$	E_2	e_1	0	E_1	E_5	E_7	0	0	0	0
$\mathbf{a_2}$	E_3	e_2	0	$-E_5$	E_1	E_6	0	0	0	0
$\mathbf{a_3}$	E_4	e_3	0	$-E_7$	$-E_6$	E_1	0	0	0	0
	E_5	e_{12}	0	0	0	0	0	0	0	0
	E_6	e_{23}	0	0	0	0	0	0	0	0
	E_7	e_{13}	0	0	0	0	0	0	0	0
	E_8	e_{123}	0	0	0	0	0	0	0	0

- GA algorithm:

```
a=a1*e1+a2*e2+a3*e3;
b=b1*e1+b2*e2+b3*e3;
?c=a*b;
```

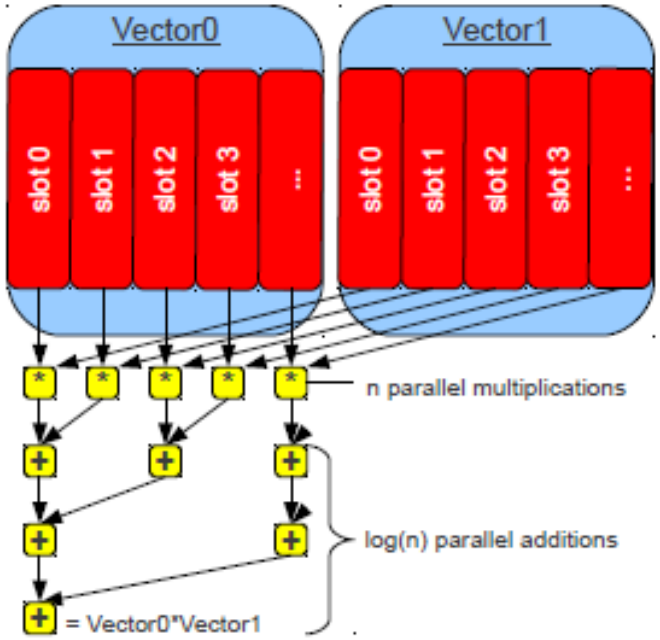
resulting C code:

```
c[1]=a1*b1+a2*b2+a3*b3;
c[5]=a1*b2-a2*b1;
c[6]=a2*b3-a3*b2;
c[7]=a1*b3-a3*b1;
```

Advantage

TECHNISCHE UNIVERSITÄT DARMSTADT

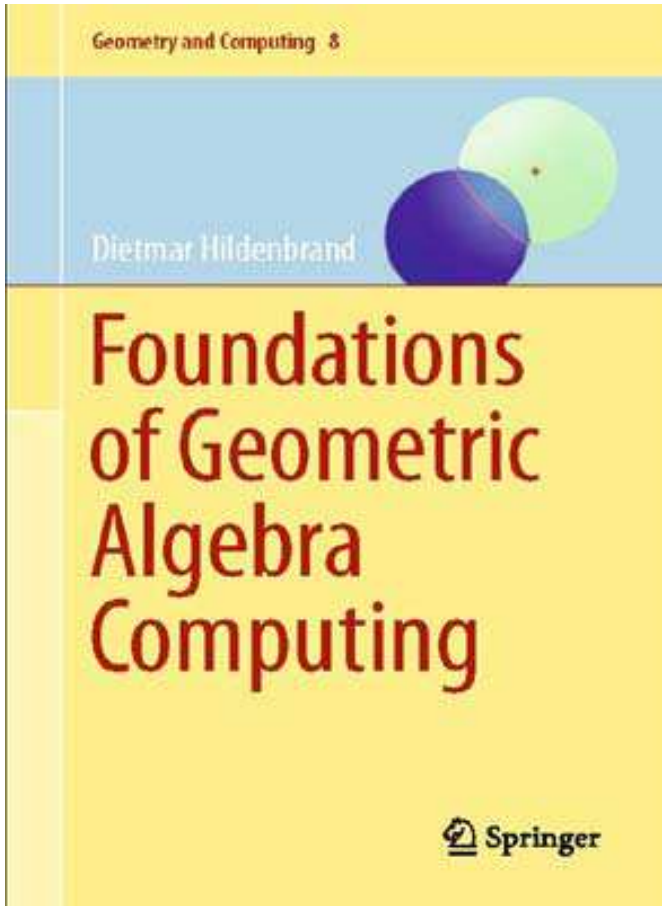
- Sum of products -> SIMD



Gaalop 2013

TECHNISCHE UNIVERSITÄT DARMSTADT

- „Foundations of Geometric Algebra Computing“
- Dietmar Hildenbrand
- Springer, 2013

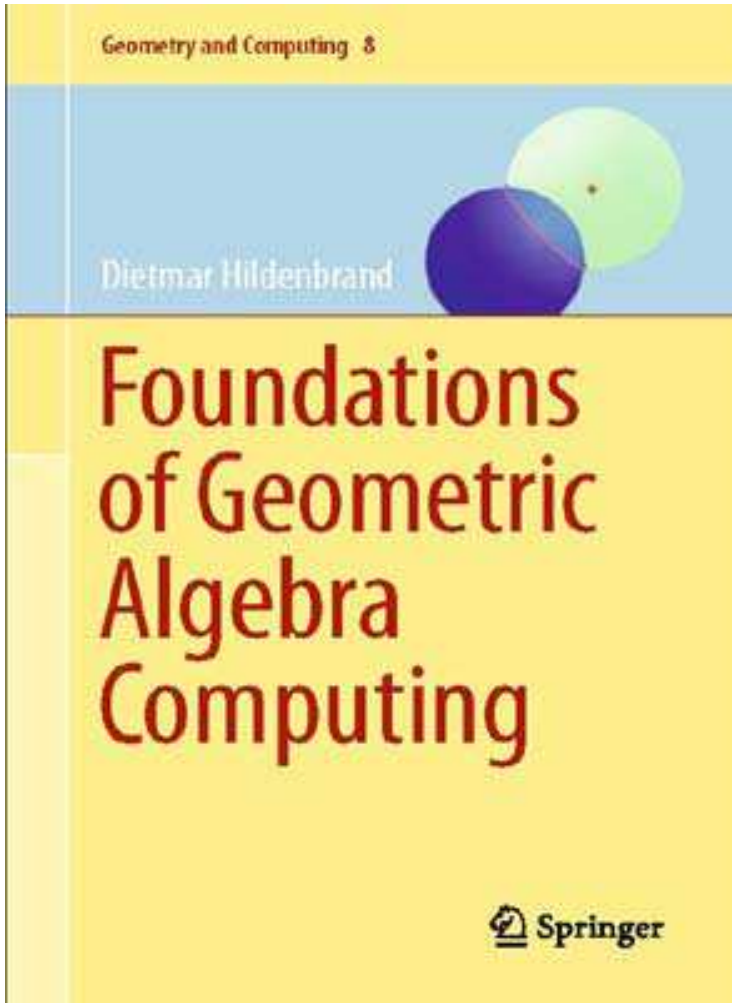


Gaalop 2013

TECHNISCHE UNIVERSITÄT DARMSTADT

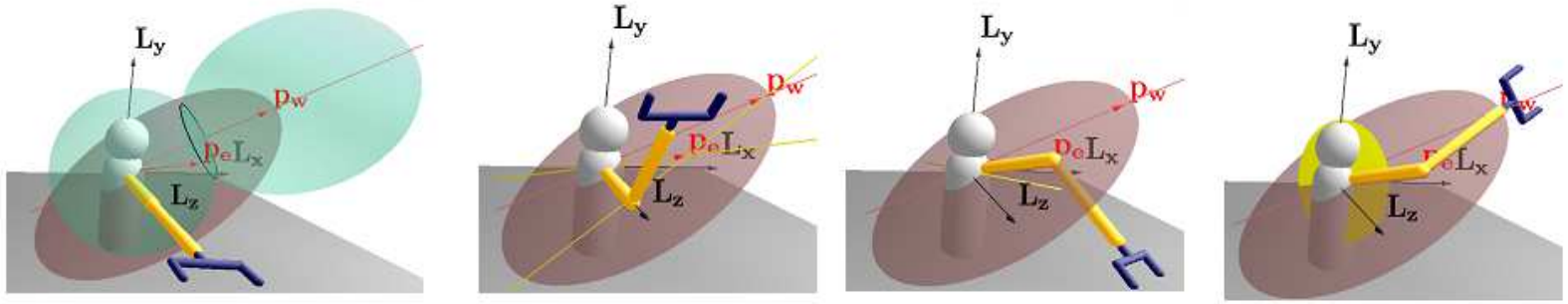
- Patrick Charrier
  - „Geometric Algebra enhanced Precompiler for C++ andOpenCL“

- Christian Steinmetz
  - Higher dimensional algebras in Gaalop
  - Visualization of multivectors
  - Fitting of high dim. objects to point clouds
  - …



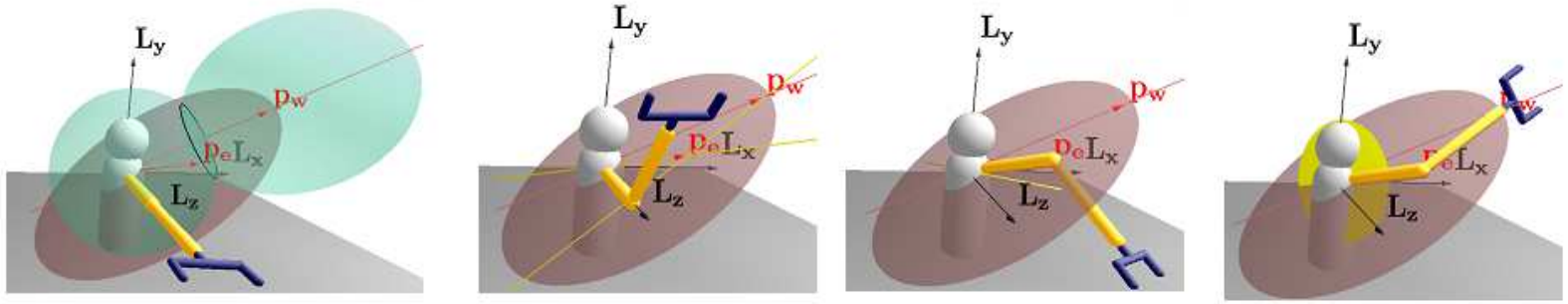
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT



- Rechnen mit geometrischer Intuitivität ✓
- Vereinheitlichung von mathematischen Systemen ✓
- Eleganz und Effizienz läßt sich vereinbaren ✓

Was bedeutet das für Algorithmen?



- Einfach zu entwickeln
- Einfach zu verstehen
- Einfach zu warten
- Effizient zu implementieren

TECHNISCHE UNIVERSITÄT DARMSTADT

vielen Dank für die Aufmerksamkeit