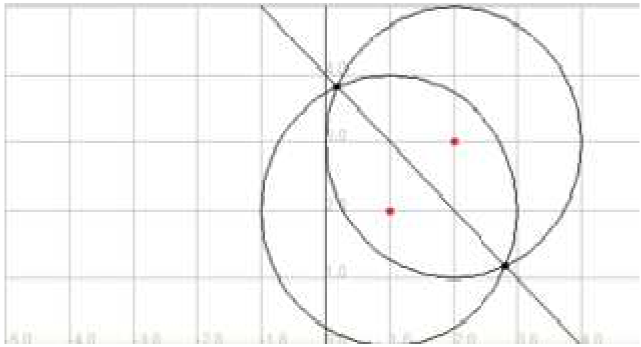
GAALOP for Different Programming Languages

May 23, 2019

Dr.-Ing. Dietmar Hildenbrand

TU Darmstadt, Germany



AGACSE conference in 2021

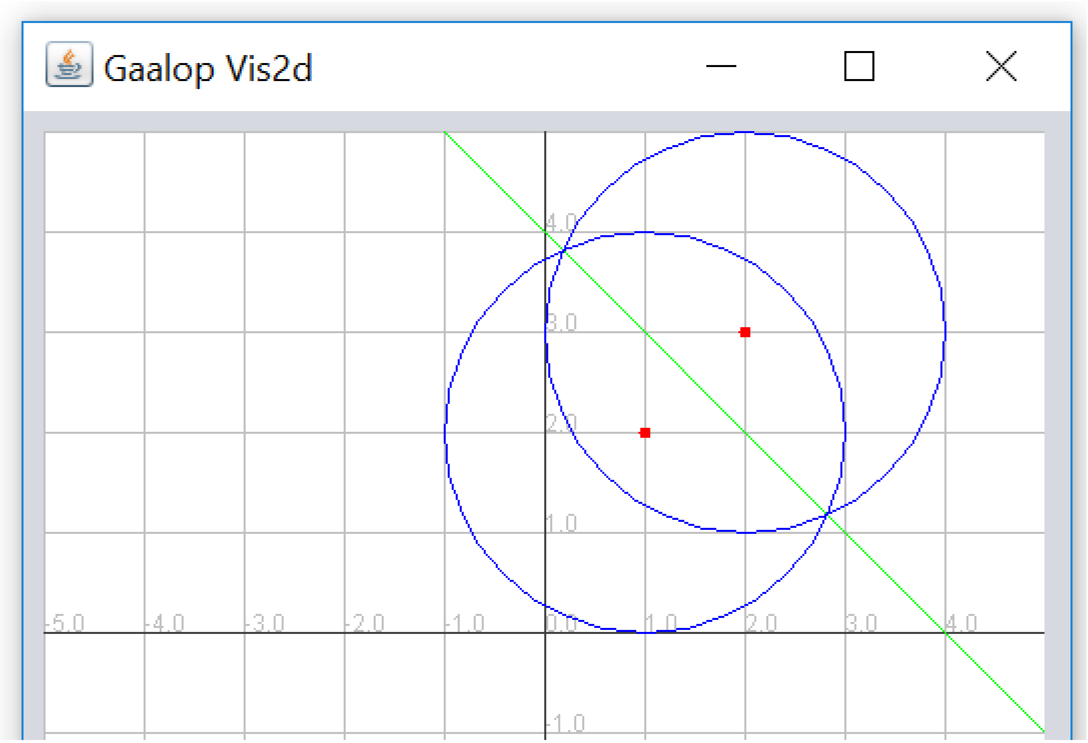
- http://agacse2021.fme.vutbr.cz/main.php

- Date: Sept. 06 – 10, 2021

Bisector Example

GAALOP Code

```
P1 = createPoint(x1,y1);
P2 = createPoint(x2,y2);
S1 = P1 - 0.5*r*r*einf;
S2 = P2 - 0.5*r*r*einf;
PP = S1^S2;
?L = *(*PP^einf);
```



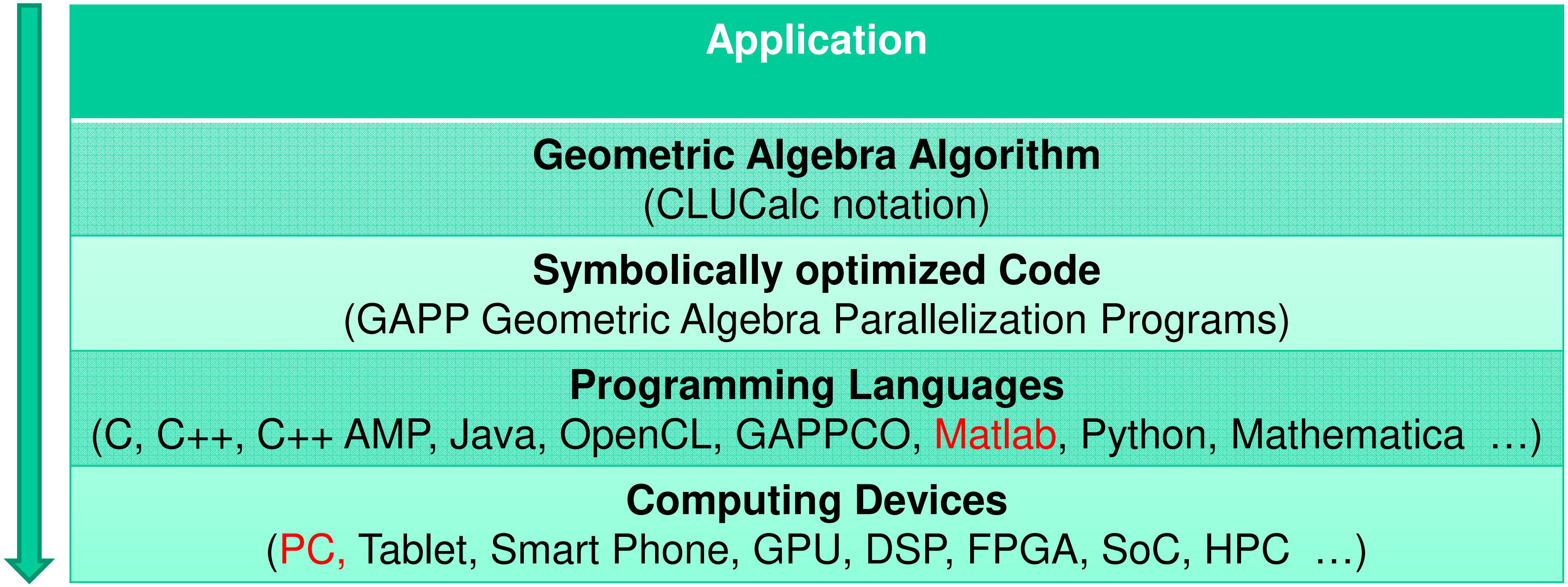
Bisector Example

Generated C-Code

```
void calculate(float x1, float x2, float y1, float y2, float L[16]) {
  L[1] = x2 - x1; // e1
  L[2] = y2 - y1; // e2
  L[3] = (y2 * y2) / 2.0 - (y1 * y1) / 2.0 + (x2 * x2) / 2.0 - (x1 * x1) / 2.0; // einf
}
```

- Only the multivector L is computed
- x1, x2, y1, y2, r are variables
- P1, P2, S1, S2, PP are only intermediate results

GAALOP -> Matlab



Application

Geometric Algebra Algorithm
(CLUCalc notation)

Symbolically optimized Code
(GAPP Geometric Algebra Parallelization Programs)

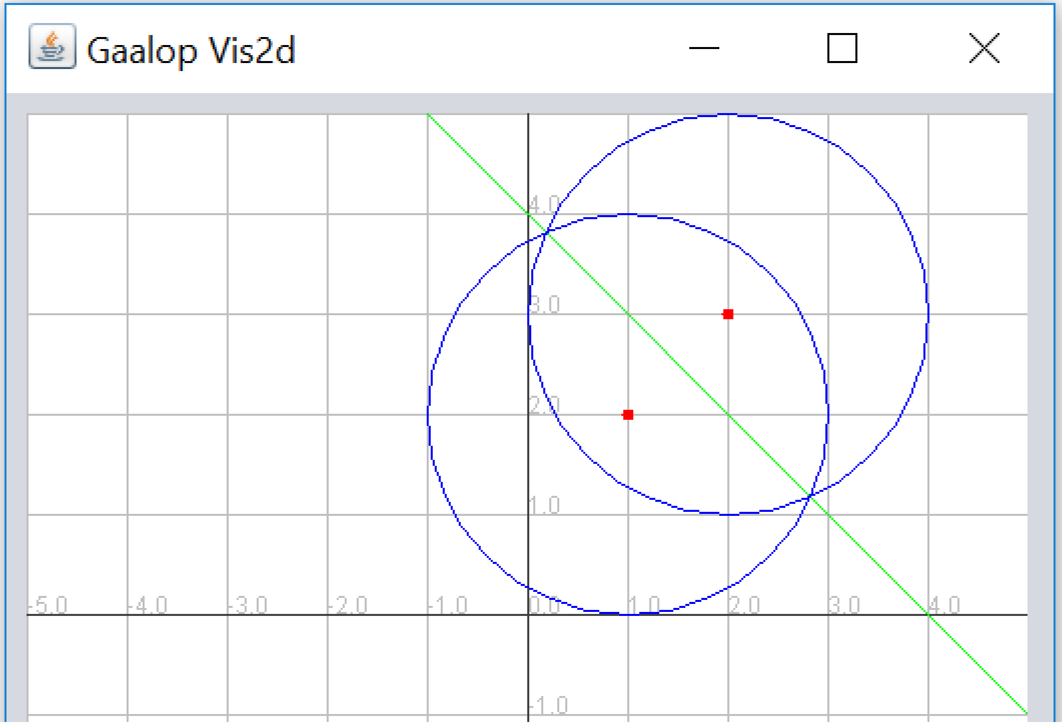
Programming Languages
(C, C++, C++ AMP, Java, OpenCL, GAPPCO, Matlab, Python, Mathematica …)

Computing Devices
(PC, Tablet, Smart Phone, GPU, DSP, FPGA, SoC, HPC …)

GAALOP -> Matlab

1. Generate C code and save it (Bisector.c)

```
P1 = createPoint(x1,y1);
P2 = createPoint(x2,y2);
S1 = P1 - 0.5*r*r*einf;
S2 = P2 - 0.5*r*r*einf;
PP = S1^S2;
?L = *(*PP^einf);
```



Bisector Example

2. Automatically import optimized Matlab Code

```
function [L] = Bisector (x1, x2, y1, y2)
        L(1) = x2 - x1;
        L(2) = y2 - y1;
        L(3) = (y2 * y2) / 2.0 - (y1 * y1) / 2.0 + (x2 * x2) / 2.0 - (x1 * x1) / 2.0;
end
```

via

```
importGAALOP(‘L’, ‘Bisector’)
```

Bisector Example

Matlab Call

```
>> L = Bisector(1,2,3,4)
L =
   1   1   5
>>
```

Only after a change of Bisector.clu in GAALOP, you have to call

```
>> importGAALOP(‘L’, ‘Bisector’)
```

again

Recall: Bisector Example

Generated C-Code

```
void calculate(float x1, float x2, float y1, float y2, float L[16]) {
  L[1] = x2 - x1; // e1
  L[2] = y2 - y1; // e2
  L[3] = (y2 * y2) / 2.0 - (y1 * y1) / 2.0 + (x2 * x2) / 2.0 - (x1 * x1) / 2.0; // einf
}
```

- Only the multivector L is computed
- x1, x2, y1, y2, r are variables

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
Line = strrep(Line, multivector, '');
```

delete the name of the multivector (L)

```
void calculate(float x1, float x2, float y1, float y2, float [16]) {
    L[1] = 2.0 * x2 - 2.0 * x1; // e1
    L[2] = 2.0 * y2 - 2.0 * y1; // e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
Line = strrep(Line, 'float ', '');
```

delete all ‘float’s

```
void calculate( x1,  x2,  y1,  y2, [16]) {
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
Line = strtrunc(Line,index(Line,', [') -1);
```

truncate starting with ‘, [‘

```
void calculate( x1,  x2,  y1,  y2
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
MatlabString = strcat(‘function [‘, multivector, ‘] = ‘, GAFunctionName,’ ‘);
Line = strrep(Line, ‘void calculate’, MatlabString );
```

Replace ‘void calculate’ by ‘function [L] = Bisector '

```
function [L] = Bisector( x1,  x2,  y1,  y2
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
Line = strcat(Line, ‘)’);
```

add ‘)’ at the end of the line

```
function [L] = Bisector( x1,  x2,  y1,  y2)
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

first line completed

importGAALOP (multivector, GAFunctionName)

For all multivector coefficients

```
Line = strrep(Line, '//', '%');   % replace characters for comments
CoeffNo = CoeffNo+1;  % CoeffNo ={1, 2, ...}
Head = strcat( multivector, '(', int2str(CoeffNo), ')' ); % generate 'multivector(No)'
Line = substr(Line, index(Line, ']')+1); % substring after ']'
Line = strcat(Head, Line);
```

```
function [L] = Bisector( x1,  x2,  y1,  y2)
        L(1) = 2.0 * x2 - 2.0 * x1; % e1
        L(2) = 2.0 * y2 - 2.0 * y1; % e2
        L(3) = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; % einf
}
```

importGAALOP (multivector, GAFunctionName)

Last line

```
fprintf(writeFile, 'end' ); % this is working for Octave and Matlab

function [L] = Bisector( x1,  x2,  y1,  y2)
        L(1) = 2.0 * x2 - 2.0 * x1; % e1
        L(2) = 2.0 * y2 - 2.0 * y1; % e2
        L(3) = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; % einf
end
```

M-file completed

showGAALOPCoefficients

Matlab Call

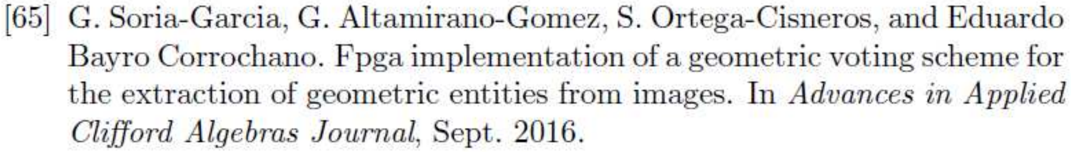
```
 >> showGAALOPCoefficients('L', 'Bisector');
1 : e1
2 : e2
3 : einf
>>
```

Based on a loop with

```
  Line = strtrim(Line); # remove leading blanks
  No = No+1;              # indices 1..
  myLine = strcat(No, " : ");
  Line = substr(Line, index(Line, "// ")+2);   # text after the comment
  Line = strcat(myLine, Line);     # concatenate index and its meaning
```

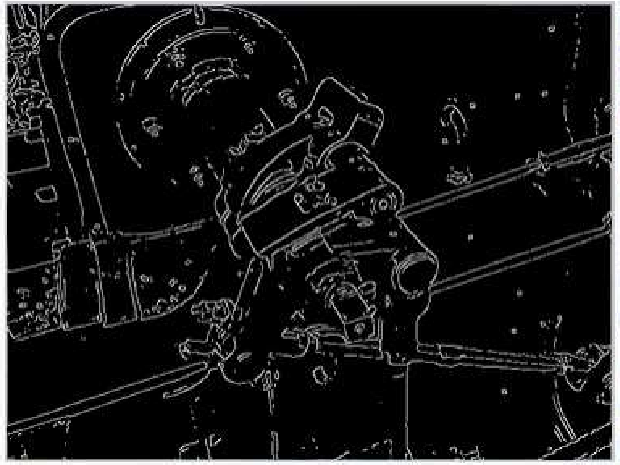
Detection of Circles and Lines in Images Using GAALOP

- CGAVS (Conformal Geometric Algebra Voting Scheme)

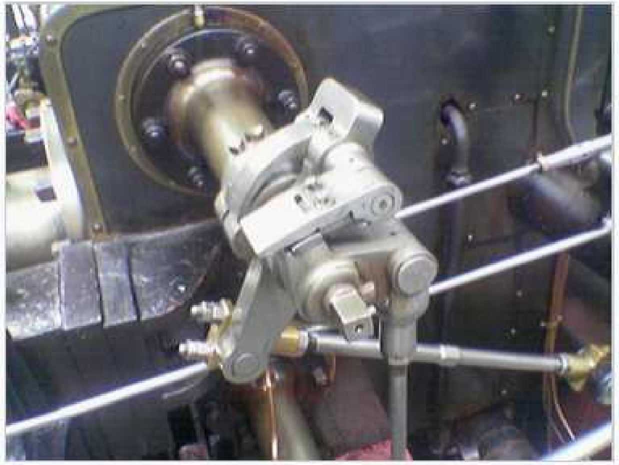


[65] G. Soria-Garcia, G. Altamirano-Gomez, S. Ortega-Cisneros, and Eduardo Bayro Corrochano. Fpga implementation of a geometric voting scheme for the extraction of geometric entities from images. In *Advances in Applied Clifford Algebras Journal*, Sept. 2016.

- Basis: edge image showing only the discontinuities of a photograph.
- https://en.wikipedia.org/wiki/Canny_edge_detector



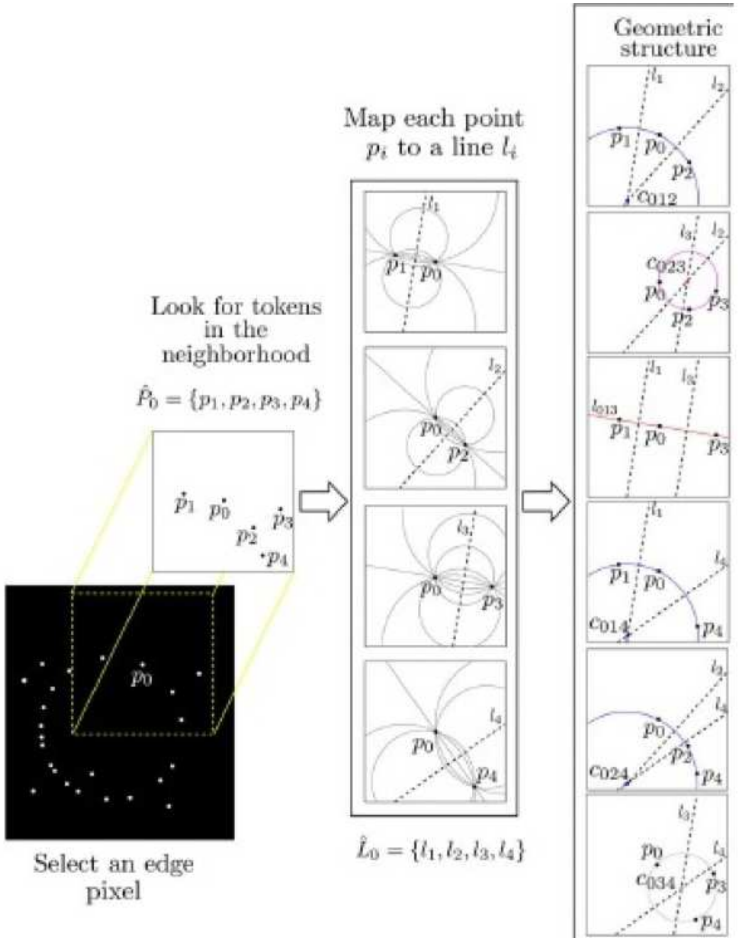
The Canny edge detector applied to a color photograph of a steam engine.



The original image.

CGAVS (Conformal Geometric Algebra Voting Scheme)

- Select an edge pixel
- Look for tokens in neighborhood
- Map each point to a line
- Intersect lines
- Circles/lines?


Select an edge pixel
Look for tokens in the neighborhood
$\hat{P}_0 = \{p_1, p_2, p_3, p_4\}$
Map each point p_i to a line l_i
$\hat{L}_0 = \{l_1, l_2, l_3, l_4\}$
Geometric structure
p_0
p_1
p_2
p_3
p_4
c_{012}
c_{023}
l_{013}
c_{014}
c_{024}
c_{034}

CGAVS (Conformal Geometric Algebra Voting Scheme)

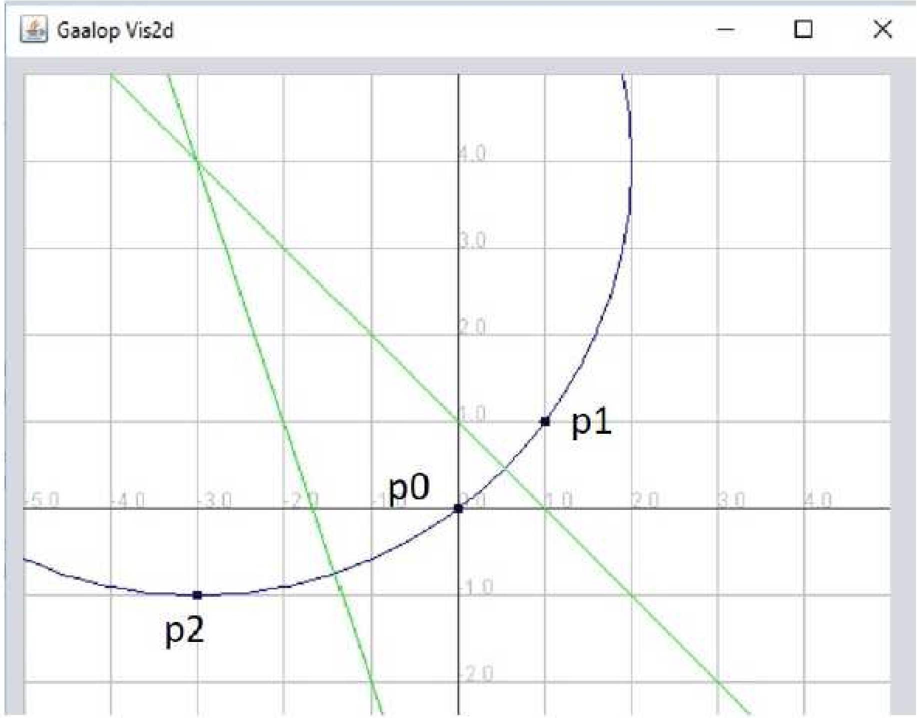
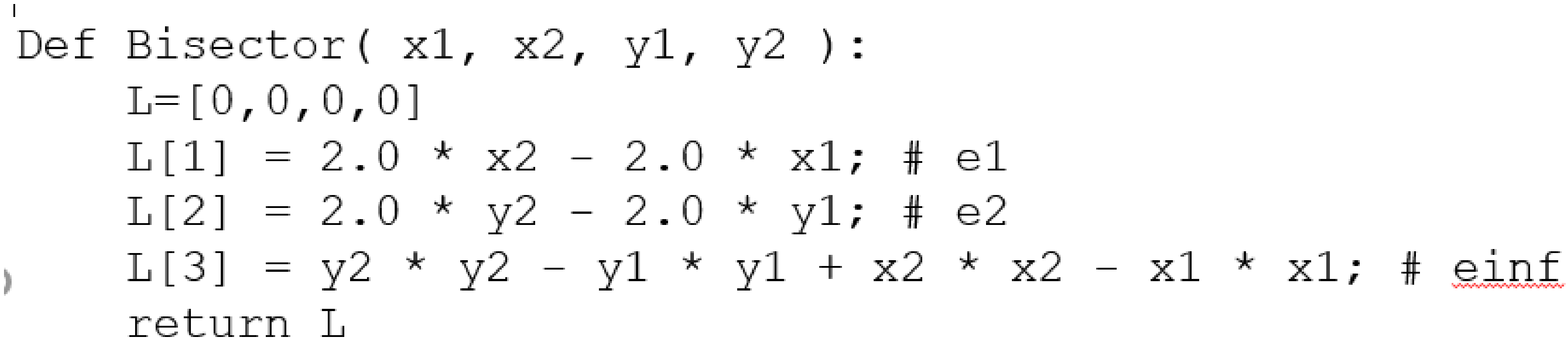


FIGURE 10.2 Visualization of *EntitiesExtraction.clu*: compute the circle through three points.

GAALOP for Python



```
Def Bisector( x1, x2, y1, y2 ):
    L=[0,0,0,0]
    L[1] = 2.0 * x2 - 2.0 * x1; # e1
    L[2] = 2.0 * y2 - 2.0 * y1; # e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; # einf
    return L
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
line = line.replace(multivector, "")
```

delete the name of the multivector (L)

```
void calculate(float x1, float x2, float y1, float y2, float [16]) {

        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
line = line.replace('float', '')
```

delete all ‘float’s

```
void calculate( x1,  x2,  y1,  y2, [16]) {
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

```
line = line.split(',  [',1)[0]
```

truncate starting with ‘, [‘

```
void calculate( x1,  x2,  y1,  y2
        L[1] = 2.0 * x2 - 2.0 * x1; // e1
        L[2] = 2.0 * y2 - 2.0 * y1; // e2
        L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Handle the first line of the C-file

line = line.replace('void calculate', ‘def ' + GAFunctionName)

line = line + ‘):‘

Complete first line in Python style

```
Def Bisector( x1, x2, y1, y2 ):

    L[1] = 2.0 * x2 - 2.0 * x1; // e1
    L[2] = 2.0 * y2 - 2.0 * y1; // e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

first line completed

importGAALOP (multivector, GAFunctionName)

Handle the second line

Initialize the multivector (according to the highest index of the multivector)

```
Def Bisector( x1, x2, y1, y2 ):
    L=[0,0,0,0]
    L[1] = 2.0 * x2 - 2.0 * x1; // e1
    L[2] = 2.0 * y2 - 2.0 * y1; // e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; // einf
}
```

importGAALOP (multivector, GAFunctionName)

Python code for the second line

```
# compute string for multivector
line = Lines[length-2] # the line with the biggest index
Split_str = line.split('[',1)
line = Split_str[1]              # text after [
Split_str = line.split(']',1)
No = Split_str[0]                # the coefficient of the multivector
InitStr = '[0'
i=1
while i<= int(No):
    InitStr += ',0'
    i+=1
InitStr += ']'
# Complete second line
SecondLine = '    ' + multivector + '=' + InitStr + '\n'
```

importGAALOP (multivector, GAFunctionName)

For all multivector coefficients

line = line.replace('//', '#')

replace characters for comments

```
Def Bisector( x1, x2, y1, y2 ):

    L[1] = 2.0 * x2 - 2.0 * x1; # e1
    L[2] = 2.0 * y2 - 2.0 * y1; # e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; # einf
}
```

importGAALOP (multivector, GAFunctionName)

Last line and definition of L

line = line + 'return ' + multivector

```
Def Bisector( x1, x2, y1, y2 ):
    L=[0,0,0,0]
    L[1] = 2.0 * x2 - 2.0 * x1; # e1
    L[2] = 2.0 * y2 - 2.0 * y1; # e2
    L[3] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; # einf
    return L
```

Python-file completed

showGAALOPCoefficients()

Python Call

```
from GAALOP import *
importGAALOP('L', 'Bisector')
showGAALOPCoefficients('L', 'Bisector')
```

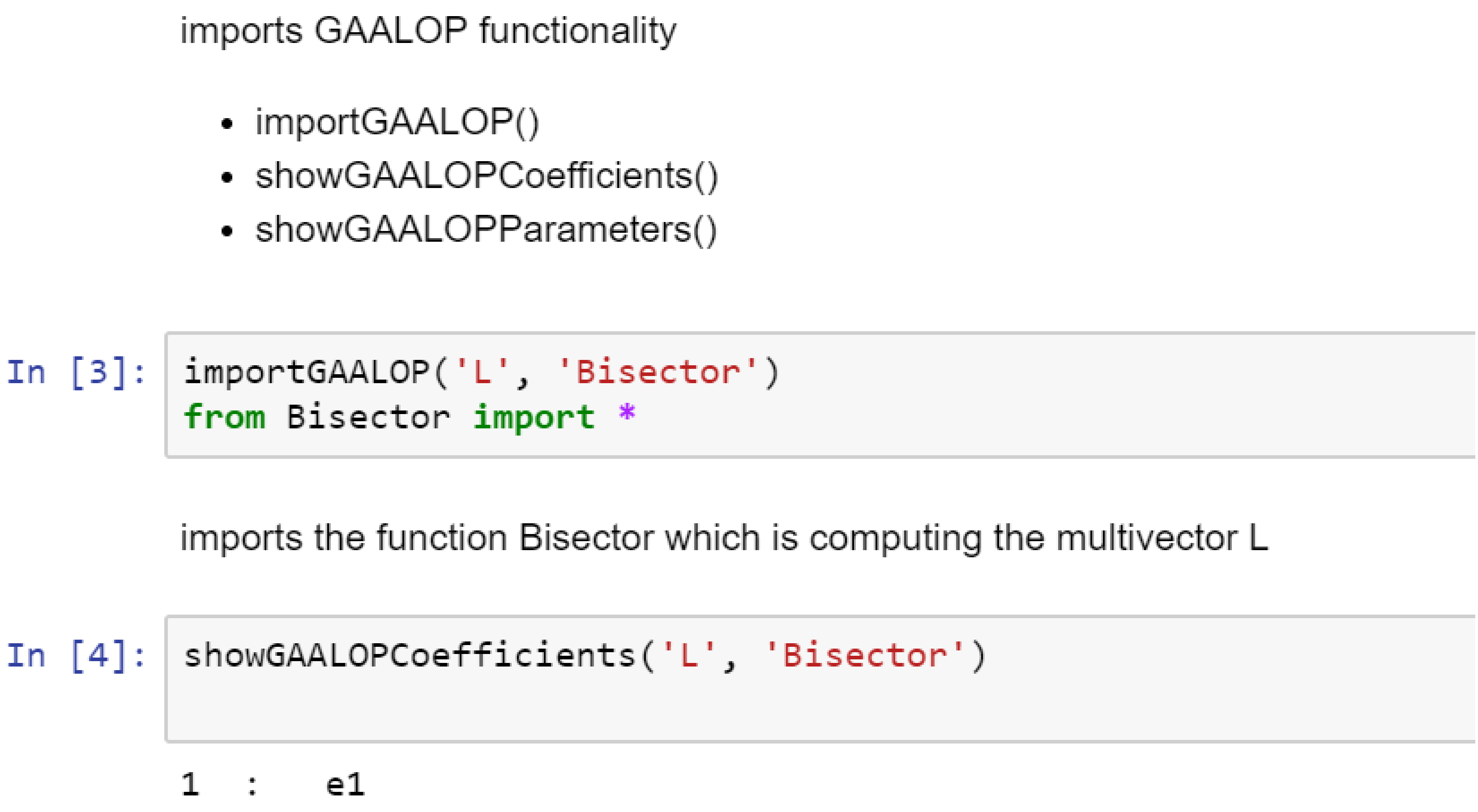
showGAALOPCoefficients()

Python Code

Loop over:

```
Split_str = line.split('[',1)
line = Split_str[1]              # text after [
Split_str = line.split(']',1)
No = Split_str[0]                # the coefficient of the multivector
line = Split_str[1]
Split_str = line.split('//',1)  # split the comment
print(No, ' : ', Split_str[1])  # index of the multivector and its meaning
```

Python Notebook



imports GAALOP functionality

- importGAALOP()
- showGAALOPCoefficients()
- showGAALOPParameters()

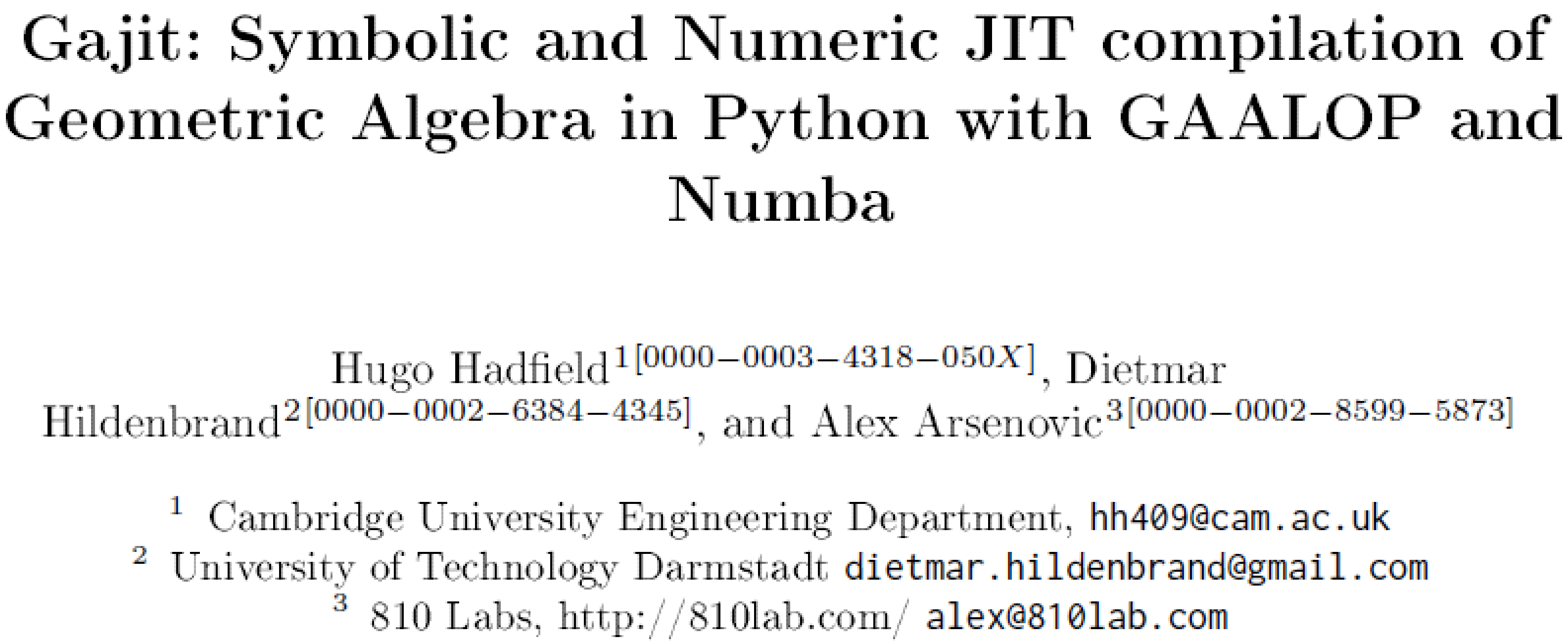
```
In [3]: importGAALOP('L', 'Bisector')
        from Bisector import *
```

imports the function Bisector which is computing the multivector L

```
In [4]: showGAALOPCoefficients('L', 'Bisector')
```

```
1  :   e1
```

Python-GAALOP Future

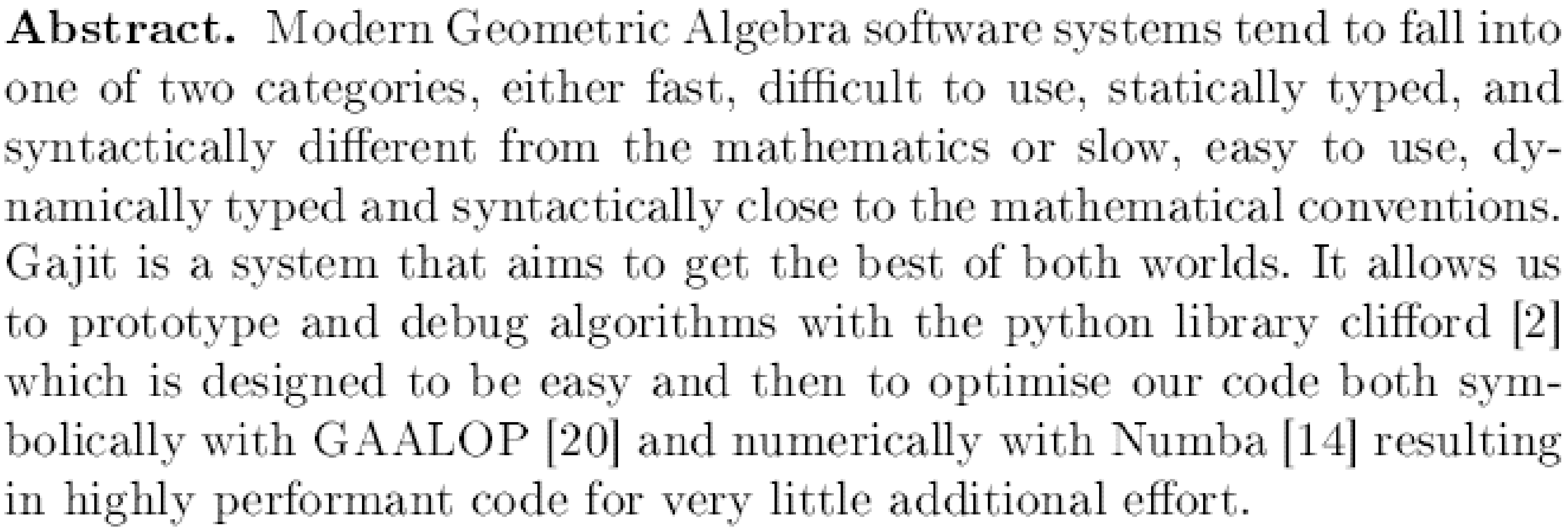


Gajit: Symbolic and Numeric JIT compilation of Geometric Algebra in Python with GAALOP and Numba

Hugo Hadfield[1][0000–0003–4318–050X], Dietmar Hildenbrand[2][0000–0002–6384–4345], and Alex Arsenovic[3][0000–0002–8599–5873]

[1] Cambridge University Engineering Department, hh409@cam.ac.uk
[2] University of Technology Darmstadt dietmar.hildenbrand@gmail.com
[3] 810 Labs, http://810lab.com/ alex@810lab.com

Python-GAALOP Future



Abstract. Modern Geometric Algebra software systems tend to fall into one of two categories, either fast, difficult to use, statically typed, and syntactically different from the mathematics or slow, easy to use, dynamically typed and syntactically close to the mathematical conventions. Gajit is a system that aims to get the best of both worlds. It allows us to prototype and debug algorithms with the python library clifford [2] which is designed to be easy and then to optimise our code both symbolically with GAALOP [20] and numerically with Numba [14] resulting in highly performant code for very little additional effort.

Bisector example in Mathematica

GAALOP-Mathematica integration just prepared for this course ☺

```
Bisector[ x1_,  x2_,  y1_,  y2_] := Module[{L},
L = ConstantArray[0,16];
L[[2]] = 2.0 * x2 - 2.0 * x1; (*e1*)
L[[3]] = 2.0 * y2 - 2.0 * y1; (*e2*)
L[[4]] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; (*einf*)
Return[{L}];]
```

- Arrays starting with 1
- Brackets for Arrays
- Specific functions such as 'sqrtf(...)', 'Sqrt[...]')

Example of Molecular Distance Geometry …

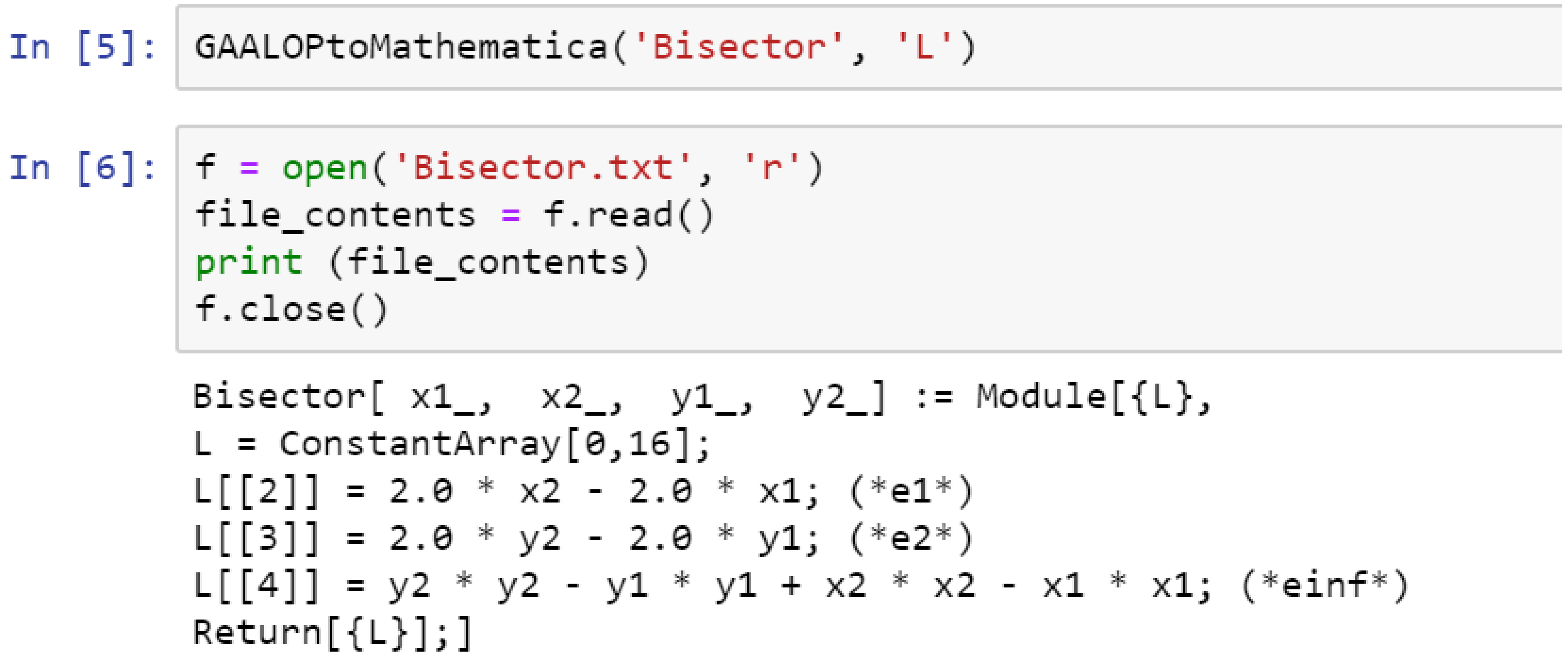
```
x1=createPoint(w1,y1,z1);
x2=createPoint(w2,y2,z2);
x3=createPoint(w3,y3,z3);

S1=x1-0.5*(d14*d14)*einf;
S2=x2-0.5*(d23*d23)*einf;
S3=x3-0.5*(d34*d34)*einf;

?PP4=S1^S2^S3;

?x4a=-(-sqrt(abs(*PP4.*PP4))+*PP4)/(einf.*PP4);
```

… transfered to Maxima …

```
ThreeSphereIntersection[ d14_, d23_, d34_, w1_, w2_, w3_, y1_, y2_, y3_,
z1_, z2_, z3_] := Module[{PP4, S1, S2, S3, x4a, x4b},
PP4 = ConstantArray[0,32];
…
x4a[[2]] = (PP4[[24]] * Sqrt[Abs[(PP4[[26]] * PP4[[26]] + PP4[[25]] * PP4[[25]] +
2.0 * PP4[[23]] * PP4[[24]] + PP4[[22]] * PP4[[22]] + 2.0 * PP4[[20]] * PP4[[21]] +
2.0 * PP4[[18]] * PP4[[19]]) - PP4[[17]] * PP4[[17]]]] - PP4[[21]] * PP4[[26]] -
PP4[[19]] * PP4[[25]] + PP4[[17]] * PP4[[24]]) / (PP4[[24]] * PP4[[24]] + PP4[[21]]
* PP4[[21]] + PP4[[19]] * PP4[[19]]); (*e1*)
…
Return[{x4a, x4b}];]
```

… by the Python notebook …

PyMathematica



```
In [5]: GAALOPtoMathematica('Bisector', 'L')
```

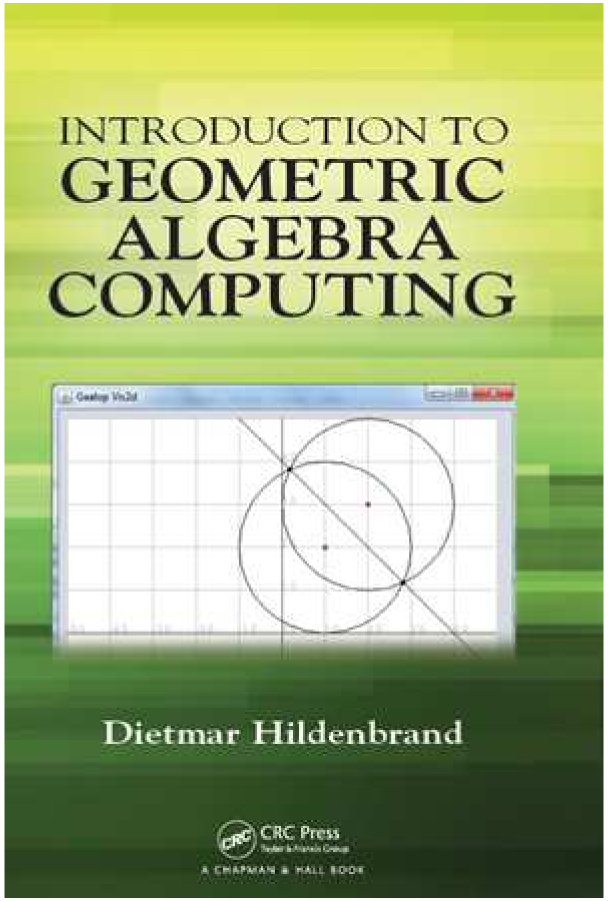
```
In [6]: f = open('Bisector.txt', 'r')
        file_contents = f.read()
        print (file_contents)
        f.close()
```

```
Bisector[ x1_,  x2_,  y1_,  y2_] := Module[{L},
L = ConstantArray[0,16];
L[[2]] = 2.0 * x2 - 2.0 * x1; (*e1*)
L[[3]] = 2.0 * y2 - 2.0 * y1; (*e2*)
L[[4]] = y2 * y2 - y1 * y1 + x2 * x2 - x1 * x1; (*einf*)
Return[{L}];]
```

Thanks a lot

Google

Dietmar Hildenbrand



INTRODUCTION TO GEOMETRIC ALGEBRA COMPUTING

Dietmar Hildenbrand

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK