Relatório do Trabalho 3 de Geometria Computacional

Bruno Dal Pontte & Ruibin Mei

2 de julho de 2025

Resumo

Esse relatório tem como objetivo explicar as escolhas de implementação e o funcionamento do Trabalho 3 de Geometria Computacional. Este trabalho consiste em implementar um algoritmo que usa uma BSP (*Binary Space Partition*) para calcular as interseções entre triêngulos e segmentos de reta. Para isso, são lidos os vértices, triângulos compostos por esses vértices, e os segmentos de reta em si. Então, os planos formados por esses triângulos são usados para subdividir o espaço em uma BSP. Ao procurar os subespaços dentro dessa BSP que contém cada extremo dos segmentos de reta, é possível obter quais planos cortam esse segmento e quais triângulos correspondem a esses planos, permitindo verificar os triângulos candidatos a interseções com cada segmento de reta de forma eficiente.

1 Introdução

1.1 Estrutura da BSP

A BSP (*Binary Space Partioning*) é uma estrutura de dados hierárquica que particiona recursivamente um espaço em subespaços usando planos divisores. Cada nó da árvore BSP gerada representa um plano que divide o espaço em dois sub planos: um na parte de frente do plano e outro a parte de trás do plano.

1.1.1 Construção da BSP

Na construção da BSP a ideia inicial foi escolher sempre o primeiro triângulo como plano divisor, mas um dos problemas disso pode ter o caso da árvore gerada ter apenas um ramo na árvore. Para tentar resolver esse problema foi escolhido a heurística de seleção aleatorizado, nesse caso aumenta a possibilidade de balancear mais a árvore gerada.[1]

1.1.2 Verificação de Interseção

A árvore BSP é utilizada na implementação com a finalidade de reduzir o número de testes de interseção necessários ao dividir recursivamente o espaço tridimensional em subespaços delimitados por planos associados aos triângulos.

A BSP por si só não realiza a detecção de interseção. Ela organiza os triângulos de forma hierárquica, permitindo que o algoritmo de busca percorra apenas as regiões relevantes onde o segmento pode intersectar algum triângulo.

Dentro de cada nó da BSP, a verificação de interseção entre o segmento e os triângulos armazenados é realizada por dois algoritmos específicos:

- O algoritmo de *Möller–Trumbore*[2] é utilizado para verificar interseção no espaço tridimensional geral.
- Quando o segmento é coplanar ao plano do triângulo, utiliza-se uma projeção ortogonal para 2D, onde são aplicados os algoritmos de teste de orientação e interseção de segmentos no plano.

Portanto, a BSP funciona como uma etapa de filtragem espacial, enquanto os testes de interseção 3D ou 2D são responsáveis por determinar efetivamente se há ou não interseção entre o segmento e os triângulos.

2 Estruturas de Dados

2.1 Estrutura PONTO

Campo	Tipo	Descrição
x	long long	Coordenada no eixo X do ponto no espaço 3D.
y	long long	Coordenada no eixo Y do ponto no espaço 3D.
z	long long	Coordenada no eixo Z do ponto no espaço 3D.

2.2 Estrutura TRIANGULO

Campo	Tipo	Descrição
vertices[3]	long long[3]	Array contendo os índices (base 1) dos três vértices que formam o triângulo.
id	long long	Identificador único do triângulo, utilizado para rastreamento.

2.3 Estrutura SEGMENTO

Campo	Tipo	Descrição
a	PONTO	Ponto inicial do segmento no espaço 3D.
b	PONTO	Ponto final do segmento no espaço 3D.

2.4 Estrutura PLANO

Campo	Tipo	Descrição
normal	PONTO	Vetor normal ao plano, representando sua orientação espacial.
distancia	long long	Distância da origem ao plano ao longo da normal.

2.5 Estrutura BSPNODO

Campo	Tipo	Descrição
planos	PLANO	Plano divisor associado ao nó da árvore BSP.
triangulos	PTRIANGULO	Lista de triângulos coplanares ao plano divisor.
qtd_triangulos	long long	Quantidade de triângulos na lista coplanar.
frente	PBSPNODO	Ponteiro para subárvore frontal em relação ao plano.
atras	PBSPNODO	Ponteiro para subárvore traseira em relação ao plano.

2.6 Estrutura ENTRADAS

Campo	Tipo	Descrição
pontos	PPONTO	Array de pontos que definem a geometria.
num_pontos	long long	Quantidade total de pontos no array.
triangulos	PTRIANGULO	Array de triângulos da malha.
num_triangulos	long long	Quantidade total de triângulos.
segmentos	PSEGMENTO	Array de segmentos a serem testados.
num_segmentos	long long	Quantidade total de segmentos.
arvoreBSP	PBSPNODO	Raiz da árvore BSP construída.

3 Algoritmo de Verificação

3.1 Verificação 3D

3.1.1 Interseção Segmento Triângulo

O objetivo é determinar se um segmento, definido por dois pontos $\mathbf{A}$ e $\mathbf{B}$, intersecta um triângulo definido pelos vértices $\mathbf{V}_0, \mathbf{V}_1, \mathbf{V}_2$.

Passo 1: Teste de Coincidência com Vértices Inicialmente, verifica-se se qualquer extremidade do segmento coincide exatamente com algum dos vértices do triângulo. Se

$$\mathbf{A} = \mathbf{V}_0 \quad \text{ou} \quad \mathbf{A} = \mathbf{V}_1 \quad \text{ou} \quad \mathbf{A} = \mathbf{V}_2$$

ou

$$\mathbf{B} = \mathbf{V}_0 \quad \text{ou} \quad \mathbf{B} = \mathbf{V}_1 \quad \text{ou} \quad \mathbf{B} = \mathbf{V}_2$$

então existe interseção (`return 1`).

Passo 2: Definição de Vetores Auxiliares

$$\begin{aligned} \mathbf{E}_1 &= \mathbf{V}_1 - \mathbf{V}_0 \quad \text{(aresta 1 do triângulo)} \\ \mathbf{E}_2 &= \mathbf{V}_2 - \mathbf{V}_0 \quad \text{(aresta 2 do triângulo)} \\ \mathbf{D} &= \mathbf{B} - \mathbf{A} \quad \text{(vetor direção do segmento)} \end{aligned}$$

Passo 3: Produto vetorial auxiliar

$$\mathbf{P} = \mathbf{D} \times \mathbf{E}_2$$

$$\mathbf{P} = (D_y E_{2z} - D_z E_{2y},\ D_z E_{2x} - D_x E_{2z},\ D_x E_{2y} - D_y E_{2x})$$

Passo 4: Cálculo do determinante

$$\det = \mathbf{E}_1 \cdot \mathbf{P}$$

$$\det = E_{1x} P_x + E_{1y} P_y + E_{1z} P_z$$

Passo 5: Verificação de paralelismo Se

$$\det = 0$$

então o segmento é paralelo ao plano do triângulo. Verifica-se se é coplanar calculando a normal do triângulo:

$$\mathbf{N} = \mathbf{E}_1 \times \mathbf{E}_2$$

$$\mathbf{N} = (E_{1y} E_{2z} - E_{1z} E_{2y},\ E_{1z} E_{2x} - E_{1x} E_{2z},\ E_{1x} E_{2y} - E_{1y} E_{2x})$$

e os produtos escalares:

$$\text{planoA} = \mathbf{N} \cdot (\mathbf{A} - \mathbf{V}_0)$$

$$\text{planoB} = \mathbf{N} \cdot (\mathbf{B} - \mathbf{V}_0)$$

Se ambos são zero, o segmento está no plano do triângulo, e o problema é reduzido para um teste 2D de interseção coplanar. Caso contrário, não há interseção.

Passo 6: Cálculo de u (parâmetro baricêntrico)

$$\mathbf{T} = \mathbf{A} - \mathbf{V}_0$$

$$u_{\text{num}} = \mathbf{T} \cdot \mathbf{P}$$

Critérios:

$$\text{se } \det > 0 \text{ ou} \quad u_{\text{num}} < 0 \text{ ou} \quad \det < u_{\text{num}}$$

$$\text{se } \det < 0 \text{ ou} \quad u_{\text{num}} > 0 \text{ ou} \quad \det > u_{\text{num}}$$

Se não satisfaz, não há interseção.

Passo 7: Cálculo de v (segundo parâmetro baricêntrico) Produto vetorial auxiliar:

$$\mathbf{Q} = \mathbf{T} \times \mathbf{E}_1$$

$$\mathbf{Q} = (T_y E_{1z} - T_z E_{1y},\ T_z E_{1x} - T_x E_{1z},\ T_x E_{1y} - T_y E_{1x})$$

Cálculo do numerador de v:

$$v_{\text{num}} = \mathbf{D} \cdot \mathbf{Q}$$

Critérios:

$$\text{se det} > 0 \text{ ou} \quad v_{\text{num}} < 0 \text{ ou} \quad \det < v_{\text{num}}$$

$$\text{se det} < 0 \text{ ou} \quad v_{\text{num}} > 0 \text{ ou} \quad \det > v_{\text{num}}$$

Além disso:

$$u_{\text{num}} + v_{\text{num}} > \det \quad \text{sedet} > 0 (\text{ou} < \det \text{ se det} \leq 0)$$

Caso contrário, o ponto está fora do triângulo.

Passo 8: Cálculo de t (parâmetro ao longo do segmento)

$$t_{\text{num}} = \mathbf{E}_2 \cdot \mathbf{Q}$$

Critérios:

$$\text{se det} > 0 \text{ ou} \quad t_{\text{num}} < 0 \text{ ou} t_{\text{num}} > \det$$

$$\text{se det} < 0 \text{ ou} \quad t_{\text{num}} > 0 \text{ ou} t_{\text{num}} < \det$$

Se não satisfaz, o ponto de interseção está fora dos limites do segmento.

Passo 9: Conclusão Se todas as condições dos passos 6, 7 e 8 forem satisfeitas, então há interseção entre o segmento e o triângulo em espaço 3D. Caso contrário, não há.

3.2 Verificação 2D (Projeção Ortogonal)

3.2.1 Interseção Coplanar

Parâmetros	**Descrição**
`A, B`	Extremidades do segmento
`V0, V1, V2`	Vértices do triângulo
`normal`	Vetor normal do plano
Saída	1 (interseção) ou 0 (sem interseção)

Formulação Matemática Quando o segmento é coplanar ao triângulo, projetamos o triângulo e o segmento para um plano 2D para realizar o teste de interseção.

Passo 1: Seleção do Eixo de Projeção

Selecionamos o eixo cuja componente do vetor normal possui maior valor absoluto:

$$\text{eixo} = \arg\max \{|\text{normal}_x|, |\text{normal}_y|, |\text{normal}_z|\}$$

O eixo selecionado é descartado, e os outros dois são usados para formar o sistema 2D.

Passo 2: Projeção dos Pontos para 2D

Se:

- **eixo** = x: projetamos para o plano yz, usando (y, z).
- **eixo** = y: projetamos para o plano xz, usando (x, z).
- **eixo** = z: projetamos para o plano xy, usando (x, y).

Denotamos os pontos projetados como:

$$\mathbf{A} = (A_x, A_y),\ \mathbf{B} = (B_x, B_y)$$

$$\mathbf{V}_0 = (V_{0x}, V_{0y}),\ \mathbf{V}_1 = (V_{1x}, V_{1y}),\ \mathbf{V}_2 = (V_{2x}, V_{2y})$$

Passo 3: Verificar se A ou B está dentro do triângulo

Usa-se o teste de orientação 2D com determinantes:

$$d_1 = (A_x - V_{1x}) \cdot (V_{0y} - V_{1y}) - (V_{0x} - V_{1x}) \cdot (A_y - V_{1y})$$
$$d_2 = (A_x - V_{2x}) \cdot (V_{1y} - V_{2y}) - (V_{1x} - V_{2x}) \cdot (A_y - V_{2y})$$
$$d_3 = (A_x - V_{0x}) \cdot (V_{2y} - V_{0y}) - (V_{2x} - V_{0x}) \cdot (A_y - V_{0y})$$

Se os sinais de d_1, d_2, d_3 forem todos positivos ou todos negativos (ou zero), então o ponto A está dentro ou na borda do triângulo.

O mesmo é feito para o ponto B.

Passo 4: Teste de Interseção com as Arestas do Triângulo

Para cada aresta do triângulo:

$$[\mathbf{V}_0, \mathbf{V}_1],\ [\mathbf{V}_1, \mathbf{V}_2],\ [\mathbf{V}_2, \mathbf{V}_0]$$

aplicamos o teste de interseção entre segmentos 2D.

Dado segmento $S_1 = [P_1, Q_1]$ e $S_2 = [P_2, Q_2]$, verifica-se:

$$o_1 = \text{orientação}(P_1, Q_1, P_2)$$
$$o_2 = \text{orientação}(P_1, Q_1, Q_2)$$
$$o_3 = \text{orientação}(P_2, Q_2, P_1)$$
$$o_4 = \text{orientação}(P_2, Q_2, Q_1)$$

Condições de cruzamento:

- Se $o_1 \neq o_2$ e $o_3 \neq o_4$, os segmentos se cruzam.
- Se qualquer orientação for zero, verifica-se se o ponto colinear está no segmento usando:

$$\min(x_1, x_2) \leq x_p \leq \max(x_1, x_2)$$
$$\min(y_1, y_2) \leq y_p \leq \max(y_1, y_2)$$

Passo 5: Resultado

O segmento intersecta o triângulo se qualquer uma das seguintes condições for verdadeira:

- O ponto A está dentro do triângulo.
- O ponto B está dentro do triângulo.
- O segmento cruza qualquer uma das arestas do triângulo.

3.3 Busca na Árvore BSP

Parâmetros	**Descrição**
`nodo`	Nó atual da árvore
`a, b`	Extremidades do segmento
`pontos`	Array de vértices
`marcado`	Array de flags de interseção
Ação	Atualiza `marcado` com triângulos intersectados

Estratégia de busca:

- Verifica interseção com triângulos no nó atual
- Classifica segmento em relação ao plano divisor
- Se cruzado ou coplanar, busca recursiva em ambos os lados
- Caso contrário, busca apenas no semi-espaço relevante

4 Casos de Teste

Os casos de testes utilizados neste trabalho foram os seguintes:

4.1 entrada.txt

O arquivo `entrada.txt` corresponde à entrada fornecida pelo professor no enunciado do trabalho.

`entrada.txt`

```
12 4 3
10 20 20
10 20 80
10 80 20
90 20 20
90 80 20
90 20 80
20 10 20
20 10 80
80 10 20
20 20 10
20 80 10
80 20 10
1 2 3
4 5 6
7 8 9
10 11 12
40 5 40 40 95 40
40 40 5 40 40 95
5 40 40 95 40 40
```

Saída correspondente:

```
1 3
1 4
2 1 2
```

4.2 diag1.in

O arquivo `diag1.in` corresponde a quatro triângulos ao redor de um único segmento de reta diagonal. Nenhum deles intersecta essa diagonal.

`diag1.in`

```
12 4 1
10 20 10
10 20 20
20 20 10
40 30 30
40 40 30
40 30 40
60 70 60
60 70 70
70 70 60
90 80 80
90 90 80
90 80 90
1 2 3
4 5 6
7 8 9
10 11 12
5 5 5 95 95 95
```

Saída correspondente:

```
0
```

4.3 diag2.in

O arquivo `diag2.in` corresponde corresponde a quatro triângulos ao redor de um único segmento de reta diagonal, similares aos da entrada anterior. Entretanto, as coordenadas deles estão trocadas de forma a passar por essa diagonal, portanto todos intersectam essa diagonal.

`diag2.in`

```
12 4 1
10 20 10
10 20 20
20 10 10
40 30 30
40 40 30
30 30 40
60 70 60
60 70 70
70 60 60
90 80 80
90 90 80
80 80 90
1 2 3
4 5 6
7 8 9
10 11 12
5 5 5 95 95 95
```

Saída correspondente:

```
4 1 2 3 4
```

4.4 plane1.in

Os arquivos `plane1.in`, `plane2.in` e `plane3.in` correspondem às entradas a seguir, onde um triângulo está ao redor de alguns segmentos de reta. Essas entradas são bem similares, sendo que em todas elas esse triângulo intersecta a maioria desses segmentos, exceto pelo quinto e último segmento. A diferença entre esses arquivos é que os eixos X, Y e Z estão transpostos de um arquivo em relação ao outro, sendo que os eixos X,Y,Z do arquivo `plane1.in` correspondem aos eixos Z,X,Y do arquivo `plane2.in` e aos eixos Y,Z,X do arquivo `plane3.in`, respectivamente.

`plane1.in`

```
3 1 5
2 2 2
2 2 11
2 11 2
1 2 3
2 2 2 2 2 3
2 2 2 2 2 2
2 2 2 2 2 11
2 2 1 2 2 11
2 2 12 2 2 16
```

`plane2.in`

```
3 1 5
 2  2  2
 2 11  2
11  2  2
```

```
1 2 3
 2  2  2   2  3  2
 2  2  2   2  2  2
 2  2  2   2 11  2
 2  1  2   2 11  2
 2 12  2   2 16  2
```

`plane3.in`

```
3 1 5
 2  2  2
11  2  2
 2  2 11
1 2 3
 2  2  2   3  2  2
 2  2  2   2  2  2
 2  2  2  11  2  2
 1  2  2  11  2  2
12  2  2  16  2  2
```

Saída correspondente às entradas `plane*.in` anteriores:

```
1 1
1 1
1 1
1 1
0
```

4.5 1.in

Nesse arquivo de entrada, alguns triângulos, que formam aproximadamente um pentágono e uma estrela de 5 pontas, passam por entre alguns segmentos de reta. A maioria dos segmentos está adiante, ou seja, fora da região dos triângulos, ou passa por eles sem intersectar os triângulos, mas o segmento de reta 2 atravessa vários dos triângulos.

`1.in`

```
5 10 5
6 23 39
85 30 68
97 69 43
61 64 36
86 48 84
4 1 5
1 5 2
1 2 4
4 5 1
5 4 1
3 5 2
2 4 1
1 3 2
5 3 4
2 1 5
85 17 60 19 29 57
31 15 35 85 70 52
44 22 26 58 19 18
80 13 12 41 22 83
44 12 98 32 77 74
```

Saída correspondente:

```
0
4 3 7 8 9
0
```

```
0
0
```

4.6 2.in

Nessa entrada, alguns triângulos estão dispostos em forma de quadriláteros que aproximam retângulos. Novamente, alguns dos segmentos, neste caso os segmentos 3 e 4, intersectam os triângulos, enquanto os demais estão para fora da região dos triângulos.

2.in

```
10 10 5
99 53 27
39 44 77
82 4 10
4 11 13
92 47 10
46 13 70
15 56 57
47 22 54
13 36 91
22 72 85
4 1 3
3 8 10
5 7 4
6 5 2
9 5 3
5 10 1
3 5 6
4 10 3
2 5 10
9 1 10
68 96 70 15 80 94
27 69 75 23 84 41
71 49 50 14 73 42
60 23 45 99 10 54
91 30 52 92 96 72
```

Saída correspondente:

```
0
0
2 9 10
1 7
0
```

4.7 3.in

Uma entrada bem mais complexa, para testar a eficiência do algoritmo. 100 triângulos estão dispostos ao longo de 10 pontos em forma de uma falha, que vagamente lembra um octaedro ou icosaedro. Nesse meio, estão dispostos 500 segmentos de reta espalhados por esse espaço de forma quase randômica. Como a entrada e a saída correspondentes são naturalmente grandes demais, apenas parte da entrada e da saída estão listadas aqui, com as partes omitidas representadas por reticências. (...)

3.in

```
10 100 500
32 34 19
3 2 67
35 49 8
59 89 40
20 94 35
```

```
74 63 56
63 37 85
67 56 14
84 37 54
42 49 70
5 2 1
5 9 2
2 7 8
6 3 5
9 7 6
9 10 5
5 7 3
6 9 4
2 3 1
3 10 8
... ... ...
19 62 53 33 69 77
16 83 28 74 95 36
27 4 17 10 28 45
4 93 3 2 82 28
69 81 93 99 89 48
67 75 5 49 69 92
12 12 67 96 4 39
73 68 58 21 8 10
6 90 56 96 42 35
16 56 48 99 41 88
... ... ...
... ... ...
```

Saída correspondente:

```
0
0
0
0
0
18 4 14 18 19 20 21 23 25 32 33 46 59 62 63 64 65 71 96
10 2 3 15 53 57 62 68 74 87 94
30 2 3 6 11 16 20 24 25 26 30 35 38 49 52 54 55 56 57 60 62 70 74 80 83 87
    88 91 92 94 96
27 4 7 14 18 19 20 21 22 23 29 32 35 42 44 48 54 59 60 62
... ... ...
... ... ...
```

4.8 4.in

Outra entrada mais complexa, semelhante à anterior, novamente para testar a eficiência do algoritmo. Neste caso, 1000 triângulos estão dispostos ao longo de 10 pontos em forma de uma falha, que vagamente lembra um octaedro ou icosaedro. Nesse meio, estão dispostos 500 segmentos de reta espalhados por esse espaço de forma quase randômica. Como a entrada e a saída correspondentes são naturalmente grandes demais, apenas parte da entrada e da saída estão listadas aqui, com as partes omitidas representadas por reticências. (...)

4.in

```
10 1000 500
68 27 64
13 27 25
95 62 98
65 75 98
80 20 31
35 86 3
78 37 5
18 68 75
63 6 67
```

```
71 53 33
7 2 8
8 10 2
2 8 9
10 2 4
1 8 2
1 8 6
6 2 4
1 7 8
10 2 5
7 8 10
... ... ...
1 62 79 37 86 72
58 6 98 15 68 45
88 18 96 45 62 73
77 60 96 41 67 47
45 59 45 91 15 95
45 18 41 46 27 55
51 57 5 7 5 53
11 40 24 25 81 51
33 3 11 91 84 17
12 25 28 93 18 92
... ... ...
... ... ...
```

Saída correspondente:

```
0
80 1 2 3 5 12 17 28 34 39 52 59 64 70 79 84 95 100 120 160 168 169 173 179
    185 196 199 203 204 221 223 233 241 251 278 283 321 324 326 328 346 367
     369 375 416 432 436 462 464 468 482 533 563 564 566 571 580 604 622
    624 627 714 715 717 724 727 765 767 769 772 786 792 793 797 798 806 894
     913 952 974 993
136 17 18 19 24 28 32 34 38 48 52 53 61 71 72 73 78 79 96 108 111 115 118
    119 140 151 152 160 164 175 199 204 209 218 219 221 233 236 246 251 259
     266 267 274 282 291 301 304 335 350 374 375 389 390 392 396 416 429
    433 435 443 446 463 464 470 471 475 481 482 484 491 506 517 519 522 524
     527 528 538 541 545 551 552 557 559 563 565 567 568 579 581 590 605
    622 624 647 662 682 683 688 693 704 705 707 708 714 728 759 769 778 782
     783 810 823 825 827 835 836 839 845 850 854 857 867 869 875 889 900
    903 917 927 931 940 954 971 975 999
191 4 7 13 25 26 32 36 37 38 61 71 73 78 96 108 111 115 116 118 119 125 129
     136 139 140 151 152 160 164 175 180 181 193 199 202 207 209 216 218
    219 221 230 234 262 266 267 282 284 290 291 301 310 327 335 338 353 354
     374 386 389 390 392 396 400 409 416 425 427 431 433 446 449 453 456
    464 470 471 475 476 482 483 484 488 489 491 495 506 507 517 522 524 527
     528 529 538 543 545 549 551 552 555 557 559 563 565 567 568 573 574
    576 579 581 584 590 591 592 605 608 615 621 624 628 635 641 645 647 655
     662 663 674 682 683 686 688 698 703 704 705 707 708 716 728 729 731
    747 757 759 769 774 778 780 782 783 803 804 810 813 820 825 827 833 836
     837 839 845 850 854 857 867 873 884 889 890 900 906 909 910 919 927
    929 931 938 940 949 954 958 968 975 980 986 999
... ... ...
... ... ...
```

Referências

[1] Geeks for Geeks. *Binary Space Partitioning*. Disponível em https://www.geeksforgeeks.org/binary-space-partitioning/. Acessado em junho de 2025.

[2] Wikipedia. *Möller–Trumbore intersection algorithm*. Disponível em https://en.wikipedia.org/wiki/M%C3%B6ller%E2%80%93Trumbore_intersection_algorithm. Acessado em junho de 2025.