Пересечение двух трёхмерных отрезков

Маняхин Дмитрий

16 июля 2024 г.

1 Определение

Отрезок в трёхмерном пространстве задан двумя точками A и B, каждая из которых имеет координаты (x, y, z). Параметрическое уравнение отрезка можно представить как:

$$\mathbf{P}(t) = \mathbf{A} + t(\mathbf{B} - \mathbf{A}), \tag{1}$$

где $0 \leq t \leq 1$ и $\mathbf{A}$ и $\mathbf{B}$ - начальная и конечная точки отрезка соответственно.

2 Параметрическое представление

Рассмотрим два отрезка в трёхмерном пространстве, заданные точками $\mathbf{P}_1$, $\mathbf{P}_2$ и $\mathbf{P}_3$, $\mathbf{P}_4$. Их параметрическое представление будет следующим:

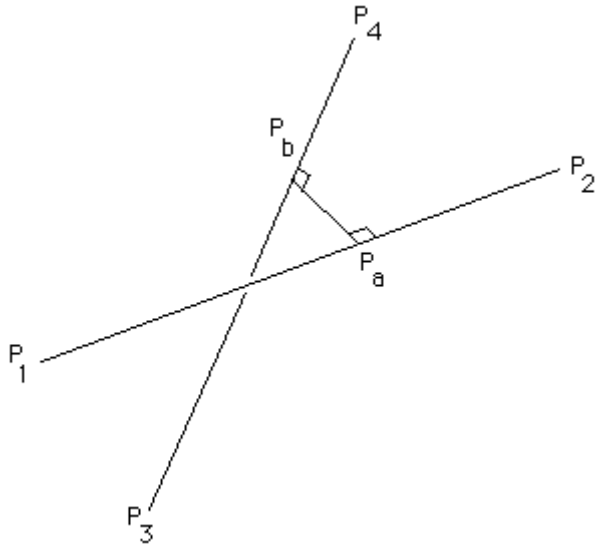
$$\mathbf{P}_a(t) = \mathbf{P}_1 + t(\mathbf{P}_2 - \mathbf{P}_1), \tag{2}$$

$$\mathbf{P}_b(s) = \mathbf{P}_3 + s(\mathbf{P}_2 - \mathbf{P}_1), \tag{3}$$

где $0 \leq t \leq 1$ и $0 \leq s \leq 1$.

3 Решение

Надежным методом является нахождение кратчайшего отрезка между двумя прямыми. Если кратчайший отрезок имеет нулевую длину, то два исходных отрезка пересекаются.



Так как кратчайший отрезок между данными отрезками будет им перпендикулярен, то:

$$(P_a - P_b)(P_2 - P_1) = 0, \tag{4}$$

$$(P_a - P_b)(P_4 - P_3) = 0, \tag{5}$$

Подставим параметрические представления:

$$(P_1 - P_3 + t(P_2 - P_1) - s(P_4 - P_3))(P_2 - P_1) = 0, \tag{6}$$

$$(P_1 - P_3 + t(P_2 - P_1) - s(P_4 - P_3))(P_4 - P_3) = 0, \tag{7}$$

Разложив полученные векторные уравнения для координат x, y и z, получим следующий результат:

$$d_{1321} + t * d_{2121} - s * d_{4321} = 0, \tag{8}$$

$$d_{1343} + t * d_{4321} - s * d_{4343} = 0, \tag{9}$$

где

$$d_{mnop} = (x_m - x_n)(x_o - x_p) + (y_m - y_n)(y_o - y_p) + (z_m - z_n)(z_o - z_p), \tag{10}$$

Найдем t и s:

$$t = (d_{1343}d_{4321} - d_{1321}d_{4343})/(d_{2121}d_{4343} - d_{4321}d_{4321}) \tag{11}$$

$$s = (d_{1343} + t * d_{4321})/d_{4343} \tag{12}$$

4 Проверка пересечения

После нахождения значений t и s необходимо проверить, лежат ли они в интервале $[0, 1]$. Если да, то точка пересечения вычисляется как:

$$\mathbf{I} = \mathbf{P}_1 + t(\mathbf{P}_2 - \mathbf{P}_1). \tag{13}$$

Если хотя бы одно из значений t или u выходит за пределы интервала $[0, 1]$, это означает, что отрезки не пересекаются в пределах своих конечных точек.

5 Алгоритм

1. Вычислить векторы направлений отрезков $\mathbf{d}_1 = \mathbf{P}_2 - \mathbf{P}_1$ и $\mathbf{d}_2 = \mathbf{P}_4 - \mathbf{P}_3$;
2. Вычислить вектор `r`, соединяющий начальные точки отрезков;
3. Вычислить параметры t и s, проверив детерминант матрицы коэффициентов (если детерминант близок к нулю, отрезки параллельны);
4. Проверить, лежат ли параметры t и s в интервале $[0, 1]$;
5. Если параметры находятся в интервале, вычислить P_a и P_b и проверить, что $P_a = P_b$.

6 Код

```
#include <iostream>
#include <optional>


class Vector3D {
public:
   double X, Y, Z;

   Vector3D(double x, double y, double z) : X(x), Y(y), Z(z) {}

   // Оператор вычитания векторов
   Vector3D operator-(const Vector3D& v) const {
      return Vector3D(X - v.X, Y - v.Y, Z - v.Z);
```

```
    }

    // Оператор сложения векторов
    Vector3D operator+(const Vector3D& v) const {
        return Vector3D(X + v.X, Y + v.Y, Z + v.Z);
    }

    // Оператор умножения вектора на скаляр
    Vector3D operator*(double scalar) const {
        return Vector3D(X * scalar, Y * scalar, Z * scalar);
    }

    // Оператор сравнения векторов
    bool operator==(const Vector3D& v) const {
        return X == v.X && Y == v.Y && Z == v.Z;
    }

    // Скалярное произведение
    double dot(const Vector3D& v) const {
        return X * v.X + Y * v.Y + Z * v.Z;
    }

    // Метод для вывода вектора
    void print() const {
        std::cout << "(" << X << ", " << Y << ", " << Z << ")\n";
    }
};


class Segment3D {
public:

    Vector3D start, end;

    // Конструктор
    Segment3D(const Vector3D& s, const Vector3D& e) : start(s), end(e) {}
};

// Функция для нахождения точки пересечения двух отрезков
std::optional<Vector3D> Intersect(const Segment3D& seg1, const Segment3D& seg2) {
    Vector3D p1 = seg1.start;
    Vector3D p2 = seg1.end;
    Vector3D p3 = seg2.start;
    Vector3D p4 = seg2.end;

    // Направляющие векторы отрезков
    Vector3D d1 = p2 - p1;
    Vector3D d2 = p4 - p3;

    if (d1.dot(d1) < std::numeric_limits<double>::epsilon() || d2.dot(d2) < std::numeric_limits<doubl
    {
        return std::nullopt;
    }

    // Вектор между начальными точками отрезков
    Vector3D r = p1 - p3;

    double d2121 = d1.dot(d1);
```

```
    double d4321 = d1.dot(d2);
    double d4343 = d2.dot(d2);
    double d1321 = d1.dot(r);
    double d1343 = d2.dot(r);

    // Вычисление знаменателя для параметров
    double denom = d2121 * d4343 - d4321 * d4321;

    // Проверка на параллельность или коллинеарность отрезков
    if (fabs(denom) < std::numeric_limits<double>::epsilon()) {
        return std::nullopt; // Отрезки параллельны или коллинеарны
    }

    double t = (d1343 * d4321 - d1321 * d4343) / denom;
    double s = (d1343 + t * d4321) / d4343;

    if (s < 0 || s > 1 || t < 0 || t > 1) {
        return std::nullopt; // Пересечения нет
    }

    // Расчёт точек кратчайшего отрезка
    Vector3D intersection = p1 + d1 * s;


    Vector3D intersection_check = p3 + d2 * t;

    // Проверка совпадения вычисленных точек пересечения
    if (intersection == intersection_check) {
        return intersection;
    }

    return std::nullopt;
}

int main() {

    // Пример использования классов и функции Intersect
    Vector3D p1(0, 0, 0);
    Vector3D p2(-1, -1, -1);
    Segment3D seg1(p1, p2);

    Vector3D p3(-1, 0, 0);
    Vector3D p4(0, -1, -1);
    Segment3D seg2(p3, p4);

    auto intersection = Intersect(seg1, seg2);
    if (intersection) {
        std::cout << "Intersection: ";
        intersection->print();
    }
    else {
        std::cout << "No intersection\n";
    }

    return 0;
}
```