МЕТОД РАСЧЕТА РАДИОЗАМЕТНОСТИ ДИСТАНЦИОННО ПИЛОТИРУЕМЫХ АППАРАТОВ ПРАВООХРАНИТЕЛЬНЫХ ОРГАНОВ

*Скотченко А.С.**

Ключевые слова: *дистанционно пилотируемый аппарат, радиозаметность, эффективная поверхность рассеяния, триангуляция, полигон, треугольник, затененный полигон, невидимый полигон, затененный треугольник, невидимый треугольник, логический обход, программная реализация.*

Аннотация.

Цель работы: *совершенствование научно-методической базы теории оценивания радиозаметности дистанционно пилотируемых аппаратов, используемых как правоохранительными органами, так и правонарушителями.*

Методы: *сравнительный анализ, вычислительная геометрия, математическая логика и компьютерная графика.*

Результаты: *исследованы особенности программных реализаций методов определения принадлежности точки полигону, используемых при расчете эффективной поверхности рассеяния дистанционно пилотируемого аппарата; проведен сравнительный анализ влияния используемых типов переменных и функций на скорость исполнения программного кода; разработан метод логического обхода вершин для определения принадлежности точки полигону, в котором применяются только арифметические операции и не вызываются функции из стандартных математических библиотек, а также используются логические переменные типа bool вместо традиционных в подобных задачах типов float и double; компьютерная реализация метода дает значительное преимущество по скорости выполнения программного кода.*

DOI: 10.21681/1994-1404-2020-4-29-37

Введение

Правоохранительные органы при решении различных задач все чаще обращаются к помощи дистанционно пилотируемых аппаратов (ДПА) [4]. В некоторых случаях, например, при проведении воздушной разведки в интересах оперативных подразделений, ДПА должен быть максимально незаметным, в том числе и для радиолокационных сигналов [9]. В существующих реалиях современной жизни остаются весьма актуальными антитеррористические задачи. Весьма полезными и важными могут быть возможность зависания ДПА над определенными объектами, возможность взять на борт высококачественную аппаратуру наблюдения, а также малая заметность аппарата [8].

Количественной мерой радиозаметности (для радиолокационных сигналов) является *эффективная* [7] *площадь рассеяния* (ЭПР) ДПА. Под ЭПР понимается площадь эквивалентного отражателя, размещенного в точке нахождения цели, который создает на приемной антенне принятый сигнал такой же интенсивности [10].

При расчете ЭПР объектов сложной формы используются численные методы, базирующиеся на предварительной триангуляции поверхности, при которой поверхность ДПА аппроксимируется треугольниками. В процессе расчетов постоянно приходится определять, какие треугольники на триангулированных поверхностях являются невидимыми и их можно исключить из дальнейших расчетов. Чем больше будет обнаружено «затененных» треугольников, тем меньше в дальнейшем придётся делать вычислений, а следовательно, существует возможность более оперативной или более точной оценки ЭПР. Последнее особенно актуально для малоразмерных летательных аппаратов, каковыми являются ДПА. При этом для каждого треугольника решается типовая *задача определения принадлежности точки треугольнику*[1] [2].

[1] Препарата Ф., Шеймос М. Вычислитель-ная геометрия: Введение. – М.: Мир, 1989. – 480 с.; Ласло М. Вычислительная гео-метрия и компьютерная графика на C++. – М.: БИНОМ, 1997. – 304 с. ISBN: 0-13-290842-5, 5-7889-0008-8; Фокс А., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. – М.: Мир, 1982. – 304 с.

* ***Скотченко Андрей Сергеевич,*** *кандидат технических наук, доцент, доцент кафедры информационного права, информатики и математики Российского государственного университета правосудия, г. Москва, Российская Федерация.*

E-mail: skotchenko@rambler.ru

```
struct Point                                    // координаты точки
        {       float x, y;             bool invisible;       };
struct Triangle             // координаты вершин треугольника
        {       Point p1, p2, p3;    };
class vector_triangle
{       public:
        vector_triangle()           // конструктор класса по умолчанию
        {x = 0;         y = 0; }
        vector_triangle(point _begin, point _end)    // конструктор
        {x = _end.x - _begin.x;             y = _end.y - _begin.y;}
}
```

Рис. 1. Листинг используемых структур и классов

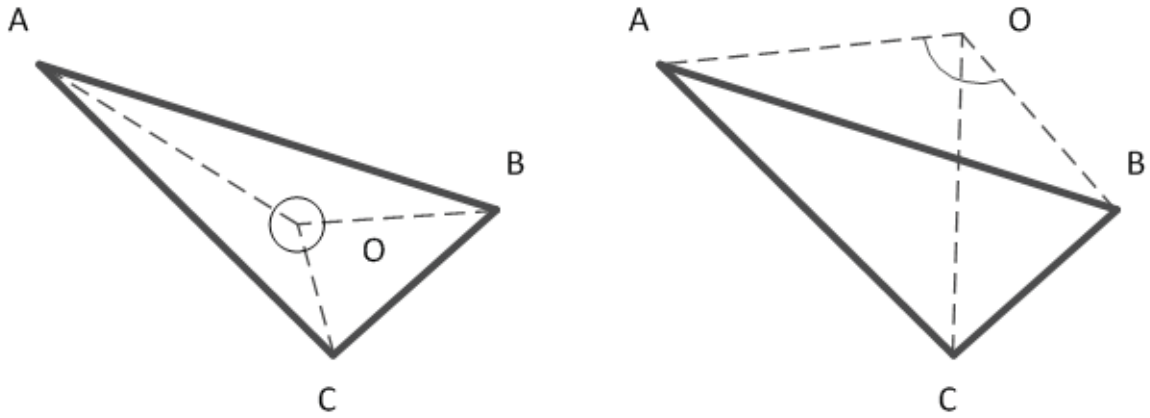

Рис. 2. Метод углов

Подобную задачу можно решить классическими методами [12], в частности, методом площадей, методом углов, методом лучей. Для сравнения известных методов и их программной реализации создадим структуры данных для точки, треугольника и класс для определения координат вектора (рис. 1).

Метод углов

В данном методе рассматриваемая точка соединяется дополнительными отрезками с вершинами треугольника[2] [1]. Для определения принадлежности точки O треугольнику ABC необходимо построить отрезки OA, OB и OC (рис. 2). Далее через скалярное произведение определяются косинусы углов с вершиной в рассматриваемой точке. Если сумма углов при точке O равна $360°$, то точка O принадлежит треугольнику ABC.

Программная реализация метода углов представлена на рис. 3.

Данный метод требует для получения приемлемых результатов переменных двойной точности типа double. Кроме того, многократный вызов функций арккосинуса acos(·) и извлечения квадратного корня sqrt(·) отрицательно сказывается на скорости выполнения программного кода.

Метод площадей (по формуле Герона)

Как и при применении метода углов, рассматриваемая точка соединяется дополнительными отрезками с вершинами треугольника[3] (рис. 2) [1]. Далее по формуле Герона через полупериметры определяются площади полученных треугольников, и их сумма сравнивается с площадью рассматриваемого треугольника.

Программная реализация метода углов представлена на рис. 4.

Данный метод, как и предыдущий, требует для получения приемлемых результатов применения переменных двойной точности и многократного вызова функции извлечения квадратного корня sqrt(·).

[2] Аммерал Л. Принципы программирования в машинной графике. М. : Сол. Систем, 1992. 224 с. ISBN 5-85316-001-6.

[3] Там же.

```
float angle(float x1, float y1, float x2, float y2)
{       return acos((x1*x2+y1*y2)/
        (sqrt(x1*x1+y1*y1)*sqrt(x2*x2+y2*y2)))*180.0/ M_PI;
}
bool method_angle (Triangle triangle, Point p)
{       double con1, con2, con3;
        con1 = angle(       triangle.p1.x - p.x, triangle.p1.y - p.y,
                            triangle.p2.x - p.x, triangle.p2.y - p.y);
        con2 = angle(       triangle.p1.x - p.x, triangle.p1.y - p.y,
                            triangle.p3.x - p.x, triangle.p3.y - p.y);
        con3 = angle(       triangle.p2.x - p.x, triangle.p2.y - p.y,
                            triangle.p3.x - p.x, triangle.p3.y - p.y);
        double sum = con1 + con2 + con3;
        if (round(sum)==360)                    return true;
        else                                    return false;
}
```

Рис. 3. Программная реализация метода углов

```
bool method_areag(Triangle triangle, point p)
{       vector_triangle ab(triangle.p1, triangle.p2);
        vector_triangle bc(triangle.p2, triangle.p3);
        vector_triangle ca(triangle.p3, triangle.p1);
        vector_triangle ao(triangle.p1, p);
        vector_triangle bo(triangle.p2, p);
        vector_triangle co(triangle.p3, p);
double perimeter_all = (ab.length() + bc.length() + ca.length())/2;
double perimeter_small_1 = (ab.length() + ao.length() + bo.length())/2;
double perimeter_small_2 = (bc.length() + bo.length() + co.length())/2;
double perimeter_small_3 = (ca.length() + co.length() + ao.length())/2;
double area_all = (sqrt((perimeter_all *
                        (perimeter_all - ab.length()) *
                        (perimeter_all - bc.length()) *
                        (perimeter_all - ca.length()))));
double area_small_1 = (sqrt(perimeter_small_1 *
                        (perimeter_small_1 - ab.length()) *
                        (perimeter_small_1 - ao.length()) *
                        (perimeter_small_1 - bo.length())));
double area_small_2 = (sqrt(perimeter_small_2 *
                        (perimeter_small_2 - bc.length()) *
                        (perimeter_small_2 - bo.length()) *
                        (perimeter_small_2 - co.length())));
double area_small_3 = (sqrt(perimeter_small_3 *
                        (perimeter_small_3 - ca.length()) *
                        (perimeter_small_3 - co.length()) *
                        (perimeter_small_3 - ao.length())));
double sum = area_small_1 + area_small_2 + area_small_3;
if (floor(area_all*rou)/rou == floor(sum*rou)/rou)      return true;
else                                                    return false;
}
```

Рис. 4. Программная реализация метода площадей (по формуле Герона)

```
float area(point a, point b, point c)
{
//      i           j           k       // Матрица
//      cx-ax       cy-ay       0       // для нахождения
//      cx-bx       cy-by       0       // векторного произведения
//      k*((cx-ax)*(cy-by) - (cx-bx)*(cy-ay)) // Определитель
return abs((c.x - a.x)*(c.y - b.y) - (c.x-b.x)*(c.y-a.y));
}
bool method_areav(Triangle abc, point a)
{
        if (area(abc.p1, abc.p2, abc.p3) ==
                area(abc.p1, abc.p2, a    ) +
                area(abc.p1, a    , abc.p3) +
                area(a    , abc.p2, abc.p3))        return true;
        else                                        return false;
}
```

Рис. 5. Программная реализация метода площадей (векторный)

Метод площадей (векторный)

Данный метод основан на утверждении, что при векторном произведении двух векторов будет получен вектор, длина которого равна площади параллелограмма, построенного на исходных векторах [3]. Как и в методе углов, рассматриваемая точка соединяется дополнительными отрезками с вершинами треугольника (см. рис. 2), далее через векторное произведение векторов, построенных на сторонах полученных треугольников, определяются их площади, а их сумма сравнивается с площадью рассматриваемого треугольника. Программная реализация векторного метода площадей представлена на рис. 5.

В данном методе можно использовать обычные переменные с плавающей точкой типа float. Для вычисления длины результирующего вектора используется функция abs(·).

Метод логического обхода

Суть метода состоит в том, что начало новой временной локальной системы координат размещается в точке, принадлежность которой треугольнику следует установить. Координаты треугольника пересчитываются в этой локальной системе координат [11]. Далее, продвигаясь по часовой стрелке от первой вершины к следующей вершине по прямым линиям, на которых расположены стороны треугольника, определяем, слева или справа от каждой прямой находится начало координат.

Если для всех сторон начало координат находится справа, то точка принадлежит треугольнику.

Запишем каноническое уравнение прямой:

$$\frac{x_n - x_{n+1}}{x_{n+2} - x_{n+1}} = \frac{y_n - y_{n+1}}{y_{n+2} - y_{n+1}} \qquad (1)$$

и представим его в виде:

$$(x_n - x_{n+1})(y_{n+2} - y_{n+1}) = \\ = (x_{n+2} - x_{n+1})(y_n - y_{n+1}); \qquad (2)$$

$$(x_n - x_{n+1})(y_{n+2} - y_{n+1}) - \\ - (x_{n+2} - x_{n+1})(y_n - y_{n+1}) = 0, \qquad (3)$$

где x_{n+1}, x_{n+2}, y_{n+1}, y_{n+2} — координаты точек, принадлежащих прямой.

Это уравнение прямой, которой принадлежит одна из сторон треугольника. Для точки, не принадлежащей прямой, выражение (3) будет больше или меньше нуля. В качестве этой точки будем рассматривать третью вершину треугольника.

Начало координат может принадлежать или не принадлежать этой прямой:

$$(-x_{n+1})(y_{n+2} - y_{n+1}) - \\ - (x_{n+2} - x_{n+1})(-y_{n+1}) = 0. \qquad (4)$$

Необходимо определить, по одну ли сторону от прямой находятся третья вершина и начало координат, т. е. одного ли знака получатся значения выражений (3) и (4). Поэтому найдем их произведение (5). Очевидно, что произведение будет больше нуля, если выражения (3) и (4) одного знака[4] [11]:

$$\big((x_n - x_{n+1})(y_{n+2} - y_{n+1}) \\ - (x_{n+2} - x_{n+1})(y_n - y_{n+1})\big)\times \\ \times\big((-x_{n+1})(y_{n+2} - y_{n+1}) - \\ (x_{n+2} - x_{n+1})(-y_{n+1})\big) > 0 \ . \qquad (5)$$

Перепишем выражение (5) в другом виде:

[4] Аммерал Л. Принципы программирования в машинной графике. М. : Сол. Систем, 1992. 224 с. ISBN 5-85316-001-6;

Фокс А., Пратт М. Вычислительная геометрия. Применение в проектировании и на производстве. М. : Мир, 1982. 304 с.

$$\left(\frac{x_n - x_{n+1}}{x_{n+2} - x_{n+1}} - \frac{y_n - y_{n+1}}{y_{n+2} - y_{n+1}}\right) \times \times \left(\frac{0 - x_{n+1}}{x_{n+2} - x_{n+1}} - \frac{0 - y_{n+1}}{y_{n+2} - y_{n+1}}\right) > 0 \quad (6)$$

Сократим запись, вычислив предварительно координаты направляющего вектора рассматриваемой прямой:

$$\vec{p}(x_{n+2} - x_{n+1}; y_{n+2} - y_{n+1}). \quad (7)$$

В итоге получим:

$$\left(\frac{x_n - x_{n+1}}{p_x} - \frac{y_n - y_{n+1}}{p_y}\right) \times \times \left(\frac{-x_{n+1}}{p_x} - \frac{-y_{n+1}}{p_y}\right) > 0 \quad (8)$$

Далее рассмотрим возможные случаи положения вершин треугольника в различных четвертях локальной системы координат. Проведем логическую замену (табл. 1).

Таблица 1

Логическая замена

координаты	замена
x<0; y<0 — координата отрицательная	x=0; y=0
x≥0; y≥0 — координата положительная	x=1; y=1

Составим таблицу истинности с учетом проведенной замены (таблица 2) [6].

Таблица 2

Таблица истинности для вершин треугольника

x_1	x_1	x_2	x_2	x_3	x_3	комментарий	значение
1	1	1	1	1	1	Все точки в I-й четверти	*ЛОЖЬ*
1	1	1	1	0	1	Две точки в I-й четверти	*ЛОЖЬ*
1	1	1	1	0	0	Две точки в I-й четверти	ИСТИНА
1	1	1	1	1	0	Две точки в I-й четверти	*ЛОЖЬ*
0	1	1	1	1	1	Две точки в I-й четверти	*ЛОЖЬ*
0	0	1	1	1	1	Две точки в I-й четверти	ИСТИНА
1	0	1	1	1	1	Две точки в I-й четверти	*ЛОЖЬ*
1	1	0	1	1	1	Две точки в I-й четверти	*ЛОЖЬ*
1	1	0	0	1	1	Две точки в I-й четверти	ИСТИНА
1	1	1	0	1	1	Две точки в I-й четверти	*ЛОЖЬ*
0	1	0	1	0	1	Все точки во II-й четверти	*ЛОЖЬ*
0	1	0	1	0	0	Две точки во II-й четверти	*ЛОЖЬ*
0	1	0	1	1	0	Две точки во II-й четверти	ИСТИНА
0	1	0	1	1	1	Две точки во II-й четверти	*ЛОЖЬ*
0	0	0	1	0	1	Две точки во II-й четверти	*ЛОЖЬ*
1	0	0	1	0	1	Две точки во II-й четверти	ИСТИНА
1	1	0	1	0	1	Две точки во II-й четверти	*ЛОЖЬ*
0	1	0	0	0	1	Две точки во II-й четверти	*ЛОЖЬ*
0	1	1	0	0	1	Две точки во II-й четверти	*ЛОЖЬ*
0	1	1	1	0	1	Две точки во II-й четверти	ИСТИНА
0	0	0	0	0	0	Все точки в III-ей четверти	*ЛОЖЬ*
0	0	0	0	0	1	Две точки во III-ей четверти	*ЛОЖЬ*
0	0	0	0	1	0	Две точки во III-ей четверти	*ЛОЖЬ*
0	0	0	0	1	1	Две точки во III-ей четверти	ИСТИНА
0	1	0	0	0	0	Две точки во III-ей четверти	*ЛОЖЬ*

Продолжение таблицы 2

x_1	x_1	x_2	x_2	x_3	x_3	комментарий	значение
1	0	0	0	0	0	Две точки во III-ей четверти	*ЛОЖЬ*
1	1	0	0	0	0	Две точки во III-ей четверти	ИСТИНА
0	0	0	1	0	0	Две точки во III-ей четверти	*ЛОЖЬ*
0	0	1	0	0	0	Две точки во III-ей четверти	*ЛОЖЬ*
0	0	1	1	0	0	Две точки во III-ей четверти	ИСТИНА
1	0	1	0	1	0	Все точки в IV-й четверти	*ЛОЖЬ*
1	0	1	0	0	0	Две точки во IV-й четверти	*ЛОЖЬ*
1	0	1	0	0	1	Две точки во IV-й четверти	ИСТИНА
1	0	1	0	1	1	Две точки во IV-й четверти	*ЛОЖЬ*
0	0	1	0	1	0	Две точки во IV-й четверти	*ЛОЖЬ*
0	1	1	0	1	0	Две точки во IV-й четверти	ИСТИНА
1	1	1	0	1	0	Две точки во IV-й четверти	*ЛОЖЬ*
1	0	0	0	1	0	Две точки во IV-й четверти	*ЛОЖЬ*
1	0	0	1	1	0	Две точки во IV-й четверти	ИСТИНА
1	0	1	1	1	0	Две точки во IV-й четверти	*ЛОЖЬ*
1	1	0	1	0	0	Все точки в разных четвертях I, II, III	ИСТИНА
1	1	0	0	0	1		ИСТИНА
0	1	1	1	0	0		ИСТИНА
0	1	0	0	1	1		ИСТИНА
0	0	1	1	0	1		ИСТИНА
0	0	0	1	1	1		ИСТИНА
1	1	0	1	1	0	Все точки в разных четвертях I, II, IV	ИСТИНА
1	1	1	0	0	1		ИСТИНА
0	1	1	1	1	0		ИСТИНА
0	1	1	0	1	1		ИСТИНА
1	0	1	1	0	1		ИСТИНА
1	0	0	1	1	1		ИСТИНА
1	1	0	0	1	0	Все точки в разных четвертях I, III, IV	ИСТИНА
1	1	1	0	0	0		ИСТИНА
1	0	1	1	0	0		ИСТИНА
1	0	0	0	1	1		ИСТИНА
0	0	1	1	1	0		ИСТИНА
0	0	1	0	1	1		ИСТИНА
0	1	0	0	1	0	Все точки в разных четвертях II, III, IV	ИСТИНА
0	1	1	0	0	0		ИСТИНА
1	0	0	0	0	1		ИСТИНА
1	0	0	1	0	0		ИСТИНА
0	0	0	1	1	0		ИСТИНА
0	0	1	0	0	1		ИСТИНА

Интерес в вышеприведенной таблице представляют только истинные значения, поскольку только в этих случаях начало координат введенной локальной системы координат может находиться внутри треугольника.

Соединим переменные в этих строках со значениями «ИСТИНА» с помощью операции конъюнкции, а сами строки между собой с помощью операции дизъюнкции. Данное выражение получается очень длинным и здесь не приводится ввиду его громоздкости.

После упрощения с помощью логических законов [5] получим:

$$x_n \,\&\, y_n \,\&\, (x_{n+2} \,\&\, y_{n+2} \vee \vee\, x_{n+1} \,\&\, y_{n+2} \vee x_{n+2} \,\&\, y_{n+1}) \,, \quad (9)$$

где $x_n, y_n, x_{n+1}, y_{n+1}, x_{n+2}, y_{n+2}$ — координаты вершин треугольника.

Рассмотрение взаимоположения точки и треугольника следует начинать именно с этого выражения. Если значение выражения (9) окажется отрицательным, то треугольник можно исключить из дальнейшего рассмотрения, так как начало координат не может оказаться внутри него. Если же значение этого выражения положительное, то следует рассмотреть взаимоположение точки и начала координат относительно сторон треугольника, используя выражения (7) и (8).

Программная реализация метода обхода вершин представлена на рис. 6 и рис. 7.

```
bool logical (bool x1, bool y1, bool x2, bool y2, bool x3, bool y3)
{
        if( x1* y1*(!x3*!y3+!x2*!y3+!x3*!y2)) return 1;
        if(!x1* y1*( x3*!y3+ x2*!y3+ x3*!y2)) return 1;
        if(!x1*!y1*( x3* y3+ x2* y3+ x3* y2)) return 1;
        if( x1*!y1*(!x3* y3+!x2* y3+!x3* y2)) return 1;
return 0;
}
```

Рис. 6. Программная реализация логической функции

```
bool method_bypass(Triangle ABC, point A)
{
        bool x1, y1, x2, y2, x3, y3, res;
        point p1, p2, p3;
// нормирование координат вершин
        p1.x = ABC.p1.x - A.x;
        p1.y = ABC.p1.y - A.y;
        p2.x = ABC.p2.x - A.x;
        p2.y = ABC.p2.y - A.y;
        p3.x = ABC.p3.x - A.x;
        p3.y = ABC.p3.y - A.y;
// применение таблицы логической замены
                p1.x>= 0 ? x1=1 : x1=0;
                p1.y>= 0 ? y1=1 : y1=0;
                p2.x>= 0 ? x2=1 : x2=0;
                p2.y>= 0 ? y2=1 : y2=0;
                p3.x>= 0 ? x3=1 : x3=0;
                p3.y>= 0 ? y3=1 : y3=0;
// вызов функции логического обхода
res = logical (x1,y1,x2,y2,x3,y3);
if (res)
        {
                if(     ((-p1.x)*(p2.y-p1.y)-(p2.x-p1.x)*(-p1.y)) *
                ((p3.x-p1.x)*(p2.y-p1.y)-(p2.x-p1.x)*(p3.y-p1.y)) < 0 )
                                                        return 0;
                if(     ((-p2.x)*(p3.y-p2.y)-(p3.x-p2.x)*(-p2.y)) *
                ((p1.x-p2.x)*(p3.y-p2.y)-(p3.x-p2.x)*(p1.y-p2.y)) < 0 )
                                                        return 0;
                if(     ((-p3.x)*(p1.y-p3.y)-(p1.x-p3.x)*(-p3.y)) *
                ((p2.x-p3.x)*(p1.y-p3.y)-(p1.x-p3.x)*(p2.y-p3.y)) < 0 )
                                                        return 0;
        }
return res;
}
```

Рис. 7. Программная реализация метода обхода вершин

Использование логической функции и её упрощение с помощью законов математической логики позволяет исключить из проверки множество треугольников. Кроме того, современные компьютеры используют двоичную систему исчисления, поэтому операции с битами переменных типа bool являются для них «родными» и выполняются максимально быстро, в отличие от операций с типами float или double. Математический аппарат аналитической геометрии применяется не для всех, а только для части треугольников и использует исключительно арифметические операции, не прибегая к вызову специальных библиотечных функций типа acos(·) или sqrt(·).

Для сравнения приведён результат работы программы при использовании различных методов (рис. 8), рассмотренных в данной статье, показывающий, что даже для небольшого количества треугольников, количество которых при формировании сцен компьютерной графики обычно исчисляется миллионами, метод логического обхода вершин оказывается наиболее быстрым.

Проверено треугольников 500 и точек 1425

Метод углов	*Время: 0.341 секунды*
Метод площадей (векторный)	*Время: 0.332 секунды*
Метод площадей (по формуле Герона)	*Время: 0.175 секунды*
Метод логического обхода	*Время: 0.038 секунды*

Рис. 8. Сравнение методов по времени выполнения

Вывод

Несомненно, использование малозаметных ДПА в деятельности правоохранительных органов будет постоянно расширяться. Кроме того, необходимость в обнаружении ДПА, используемых правонарушителями, также остается актуальной. Поэтому потребность в оперативной теоретической оценке их радиозаметности в скором времени окажется весьма востребованной. Программа логического определения принадлежности точки треугольнику методом обхода его вершин демонстрирует преимущества использования методов математической логики по сравнению с традиционными классическими методами высшей математики. Даже для небольшого количества точек и треугольников метод логического обхода вершин оказывается самым быстрым. Это преимущество достигается за счет исключения из сравнения части треугольников путём применения логической формулы, а также за счет использования только арифметических операций, логических функций и переменных.

Литература

1. Андреева Е. В., Егоров Ю. Е., Взаимное расположение точек и фигур // Информатика. 2002. № 40. С. 28—31.
2. Де Берг Марк, Чеонг Отфрид, Ван Кревельд Марк, Овермарс Марк. Вычислительная геометрия. Алгоритмы и приложения. М. : ДМК-Пресс, 2017. 437 с.
3. Ивановский С. А., Симончик С. К. Алгоритмы вычислительной геометрии. Пересечение отрезков: метод заметания плоскости // Компьютерные инструменты в образовании. СПб. : Изд-во ЦПО «Информатизация образования». 2007. № 4. С. 18—33.
4. Канушкин С. В. Синергетический подход в управлении группой беспилотных летательных аппаратов интеллектуальной системы охранного мониторинга // Правовая информатика. 2018. № 3. С. 25—37. DOI: 10.21681/1994-1404-2018-3-25-37.
5. Клини С. К. Математическая логика. Изд. 4-е. М. : Изд-во ЛКИ, 2008. 480 с. ISBN 978-5-382-00626-0.
6. Королев В. Т., Ловцов Д. А., Радионов В. В. Системный анализ. Часть. 2. Логические методы / Под ред. Д. А. Ловцова. М. : Росс. гос. ун-т правосудия, 2017. 160 с. ISBN 978-5-93916-636-6.
7. Ловцов Д. А. Информационная теория эргасистем : тезаурус. М. : Наука, 2005. 248 с. ISBN 5-02-033779-X.
8. Ловцов Д. А., Гаврилов Д. А. Моделирование оптико-электронных систем дистанционно пилотируемых аппаратов : монография. М. : Технолоджи-3000, 2019. 164 с. ISBN 978-5-94472-036-8.
9. Ловцов Д. А., Черных А. М. Геоинформационные системы. М. : Росс. акад. правосудия, 2012. 188 с. ISBN 978-5-93916-340-8.
10. Макаренко С. И., Тимошенко А. В., Васильченко А. С. Анализ средств и способов противодействия беспилотным летательным аппаратам. Часть 1. Беспилотный летательный аппарат как объект обнаружения и поражения // Системы управления, связи и безопасности. 2020. № 1. С. 109—146.
11. Роджерс Д., Адамс Дж. Математические основы машинной графики. М. : Мир, 2001. 604 с. ISBN 5-03-002143-4.
12. Скотченко А. С. Метод логического определения принадлежности точки треугольнику при построении 3D-модели // Вестник Университета Российской академии образования. 2014. № 2. С. 126—142.

Рецензент: ***Сухов Андрей Владимирович,*** *доктор технических наук, профессор, профессор кафедры радиоэлектроники, телекоммуникаций и нанотехнологий Московского авиационного института (национального исследовательского университета), Российская Федерация, Москва.*
E-mail: avs57@mail.ru

A METHOD FOR COMPUTING THE RADIO VISIBILITY OF REMOTELY PILOTED AIRCRAFT OF LAW ENFORCEMENT AGENCIES

Andrei Skotchenko, *Ph.D. (Technology), Associate Professor at the Department of Information Technology Law, Informatics and Mathematics of the Russian State University of Justice, Moscow, Russian Federation.*
E-mail: *skotchenko@rambler.ru*

Keywords: *remotely piloted aircraft, radio visibility, radar cross-section, triangulation, polygon, triangle, shaded polygon, invisible polygon, shaded triangle, invisible triangle, logical traversal, software implementation.*

Abstract.
Purpose of the work: *improving the methodological basis of the theory of evaluating the radio visibility of remotely piloted aircraft used by law enforcement agencies as well as offenders.*
Methods used: *comparative analysis, computational geometry, mathematical logic and computer graphics.*
Results obtained: *features of software implementations of methods for determining whether a point belongs to a polygon used in computing the radar cross-section of a remotely piloted aircraft are studied, a comparative analysis of the impact of types of used variables and functions on software code execution speed is carried out, and a method of logical traversal of vertices for determining whether a point belongs to a polygon which uses only arithmetic operations, bool variables instead of float and double ones traditionally used for such problems, and does not call functions from standard mathematical libraries is worked out. A software implementation of the method gives a significant gain in code execution speed.*

References

1. Andreeva E. V., Egorov Iu. E., Vzaimnoe raspolozhenie tochek i figur. Informatika, 2002, No. 40, pp. 28-31.
2. De Berg Mark, Cheong Otfrid, Van Krevel'd Mark, Overmars Mark. Vychislitel'naia geometriia. Algoritmy i prilozheniia. M. : DMK-Press, 2017, 437 pp.
3. Ivanovskii S. A., Simonchik S. K. Algoritmy vychislitel'noi geometrii. Peresechenie otrezkov: metod zametaniia ploskosti. Komp'iuternye instrumenty v obrazovanii. SPb. : Izd-vo TsPO "Informatizatsiia obrazovaniia", 2007, No. 4, pp. 18-33.
4. Kanushkin S. V. Sinergeticheskii podkhod v upravlenii gruppoi bespilotnykh letatel'nykh apparatov intellektual'noi sistemy okhrannogo monitoringa. Pravovaia informatika, 2018, No. 3, pp. 25-37. DOI: 10.21681/1994-1404-2018-3-25-37.
5. Klini S. K. Matematicheskaia logika. Izd. 4-e. M. : Izd-vo LKI, 2008, 480 pp. ISBN 978-5-382-00626-0.
6. Korolev V. T., Lovtsov D. A., Radionov V. V. Sistemnyi analiz. Chast'. 2. Logicheskie metody. Pod red. D. A. Lovtsova. M. : Ross. gos. un-t pravosudiia, 2017, 160 pp. ISBN 978-5-93916-636-6.
7. Lovtsov D. A. Informatsionnaia teoriia ergasistem : tezaurus. M. : Nauka, 2005, 248 pp. ISBN 5-02-033779-X.
8. Lovtsov D. A., Gavrilov D. A. Modelirovanie optiko-elektronnykh sistem distantsionno pilotiruemykh apparatov : monografiia. M. : Tekhnolodzhi-3000, 2019, 164 pp. ISBN 978-5-94472-036-8.
9. Lovtsov D. A., Chernykh A. M. Geoinformatsionnye sistemy. M. : Ross. akad. pravosudiia, 2012, 188 pp. ISBN 978-5-93916-340-8.
10. Makarenko S. I., Timoshenko A. V., Vasil'chenko A. S. Analiz sredstv i sposobov protivodeistviia bespilotnym letatel'nym apparatam. Chast' 1. Bespilotnyi letatel'nyi apparat kak ob"ekt obnaruzheniia i porazheniia. Sistemy upravleniia, sviazi i bezopasnosti, 2020, No. 1, pp. 109-146.
11. Rodzhers D., Adams Dzh. Matematicheskie osnovy mashinnoi grafiki. M. : Mir, 2001, 604 pp. ISBN 5-03-002143-4.
12. Skotchenko A. S. Metod logicheskogo opredeleniia prinadlezhnosti tochki treugol'niku pri postroenii 3D-modeli. Vestnik Universiteta Rossiiskoi akademii obrazovaniia, 2014, No. 2, pp. 126-142.