УДК 004.031.43, 004.93

**В.А. БОРОДИН**

# БЫСТРЫЙ МЕТОД ПОВОРОТА ИЗОБРАЖЕНИЯ СЛОЖНОГО СИМВОЛА, ВЫВОДИМОГО НА ЭКРАН ГЕОИНФОРМАЦИОННЫХ КОМПЛЕКСОВ РЕАЛЬНОГО ВРЕМЕНИ

***Abstract:*** *In the article the quick method for turn of the raster image of complex symbol that requires less then two operations of addition for each pixel of the image is shown.*
***Key words:*** *quick turn of image methods.*

***Анотація:*** *У статті розглянуто алгоритм перетворення повороту складного растрового зображення символу, що забезпечує виконання цієї операції для кожного пікселя зображення за менш, ніж дві арифметичні операції додавання.*
***Ключові слова:*** *швидкий метод повороту зображень.*

***Аннотация:*** *В статье рассмотрен алгоритм преобразования поворота сложного растрового изображения символа, который обеспечивает выполнение этой операции для каждого пикселя изображения за менее, чем две арифметических операции сложения.*
***Ключевые слова:*** *быстрый метод поворота изображений.*

## 1. Введение

Изучаемые в данной статье геоинформационные комплексы реального времени (ГК РВ) – это системы, обеспечивающие безопасность полетов, тренажеров различных транспортных средств (самолеты, космические корабли, автомобили и др.), суть которых состоит в отображении на фоне изображения карты быстропротекающих процессов, например, воздушной обстановки в виде динамических сцен.

Типичным примером применения такого комплекса в гражданской авиации может быть комплекс, содержащий систему диспетчерской службы и предназначенный для отображения летящих самолетов в районе крупного аэропорта и принятия оперативных заключений при решении задачи безопасной разводки самолетов [1–6]. Участившиеся случаи авиакатастроф сами говорят об актуальности широкого распространения таких комплексов для обеспечения безопасности полетов.

Одной из наиболее характерных для геоинформационных комплексов реального времени (ГК РВ) является задача принятия решений на основе анализа поведения некоторого множества подвижных объектов, двигающихся в околоземном пространстве [1–5].

Главной особенностью функционирования таких комплексов является оперативное отображение информации о подвижных объектах, представляемых определенными графическими обозначениями в виде достаточно сложных символов на картографическом фоне в реальном масштабе времени. Под сложным символом понимается его изображение любой конфигурации с размерами более чем 8x8 пикселей [3]. Таким символом могут быть и части фона, т.е. участок карты, на которой отображается обстановка.

## 2. Постановка задачи

Для обеспечения адекватного восприятия динамической обстановки оператором ГК РВ необходимо преодолеть трудности, возникающие при передаче плавности движения изображений, зависящие от увеличения скорости отображения, которая, в свою очередь, зависит от времени выполнения

аффинных преобразований – параллельного переноса, поворота и изменения масштаба [2, 5]. При организации отображений графических символов в реальном времени важно производить такие операции как можно быстрее, чтобы освободить процессор для решения других задач. При этом изображения символов не должны менять свою форму в зависимости от направления движения. Эти ограничения присущи задаче поворота изображения, которая является главной и наиболее сложной и при использовании известных методов занимает много вычислительных ресурсов и времени [2–5].

Предлагаемый нами метод поворота опишем на конкретном примере.

Сначала рассмотрим случай, когда изображение символа представляется в растровом виде и необходимо изменить его ориентацию. Для этого требуется произвести для каждой точки изображения геометрическую операцию поворота относительно заданной базовой точки. То есть для каждого пикселя экрана цвета Ц с координатами $(x, y)$ предполагается осуществить поворот на угол $\varphi$ с помощью синусно-косинусного преобразования [2] и определить, в какой новый пиксель экрана $(x', y')$ этот пиксель переместился, и после этого присвоить пикселю $(x', y')$ цвет Ц.

Недостатком этого метода является то, что при таком методе в новой матрице символа остаются пиксели, значение которым не присвоено.

Для решения этой задачи нами предлагается решение от обратного, т.е. для каждой точки из нового разбиения экрана с координатами $(x'', y'')$ надо осуществить поворот на $-\varphi$ и определить, в пиксель какого цвета попала эта точка, и присвоить точке с координатами $(x'', y'')$ этот цвет.

Синусно-косинусное преобразование поворота символа в матричной форме описывается формулой

$$|X^*| = |X| \cdot |T|, \tag{1}$$

где $|X^*| = (x^*, y^*, 1)$ – результирующий вектор координат точки;

$|X| = (x, y, 1)$ – вектор исходных координат точки;

$$|T| = \begin{vmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ \Delta X & \Delta Y & 1 \end{vmatrix}$$ – матрица преобразования.

Формула (1) может быть представлена в виде системы уравнений (2):

$$\begin{cases} x^* = x\ \cos\varphi - y\ \sin\varphi + \Delta X; \\ y^* = x\ \sin\varphi + y\ \cos\varphi + \Delta Y. \end{cases} \tag{2}$$

Анализ данной системы уравнений указывает на то, что для изменения координат одной точки изображения необходимо выполнить четыре операции умножения и четыре операции арифметического сложения для каждой точки растра.

Альтернативный метод преобразования изображений (метод базовых символов) рассматривался в [3]. Его основное содержание заключается в том, что в видеопамяти хранятся заранее подготовленные изображения, повернутые с шагом $22{,}5^0$. Этот метод позволяет осуществлять повороты только за счет изменений направлений считываний матрицы растрового

изображения символа, но зато требует дополнительных ресурсов памяти для хранения этих заранее подготовленных изображений.

Другие описанные в литературе [2–6] методы зачастую используют “в лоб” преобразование по формулам (2), которое позволяет выполнить поворот за $4S$ операций умножения и $4S$ операций сложения, или более простое преобразование (1) в полярной системе координат, что сокращает время выполнения до $2S$ операций умножения и $3S$ операций сложения, при условии, что $S$ – количество пикселей символа, а значение синуса и косинуса угла поворота уже предварительно вычислено. Для определенных типов простых изображений (символов), которые представляют собой композицию некоторых линий (векторов), количество арифметических операций можно сократить до $4S$ ($2S$ операций умножения и $2S$ операций сложения) [4, 5], что для некоторых приложений является вполне приемлемым.

В предлагаемом методе операция поворота символа, заданного прямоугольной матрицей $S[i, j]$, например, изображения участка карты на экране, осуществляется за 2 арифметические операции сложения для каждого пикселя. Этот метод является модификацией метода базовых символов [3] и годится как для произвольного растрового, так и произвольного векторного изображения символа.

### 3. Быстрый метод поворота

Назовем абсолютной системой координат координатную сетку, единичные клетки которой совпадают с пикселями экрана. Относительной системой координат назовем координатную сетку, единичные клетки которой совпадают с пикселями матрицы символа $S[i, j]$.

Выберем точку начала относительной системы координат $(x_1, y_1)$. Для удобства вычислений можно взять одну из вершин прямоугольной матрицы, в которой хранится растровое изображение символа. Для этой точки вычисляются ее координаты $(x', y')$ после поворота с помощью синусно-косинусного преобразования (2):

$$x' = x_1 + (x_0 - x_1)\cdot\cos(\varphi) + (y_1 - y_0)\cdot\sin(\varphi);$$
$$y' = y_1 + (y_0 - y_1)\cdot\cos(\varphi) + (x_0 - x_1)\cdot\sin(\varphi).$$

Здесь $\varphi$ – угол поворота; $x_0, y_0$ – координаты точки, относительно которой совершается поворот О.

Затем вычисляются проекции границ единичной клетки относительной системы координат на абсолютную:

$$d\times 1 = h1\cdot\sin(\varphi);\ dy1 = h1\cdot\cos(\varphi);$$
$$d\times 2 = -h2\cdot\cos(\varphi);\ dy2 = -h2\cdot\sin(\varphi).$$

Здесь $h1$, $h2$ – длина сторон пикселя.

Далее от вычисленных значений $x'$ и $y'$ новые значения вычисляются с помощью цикла по клеткам относительной системы координат. Так определяется новый цвет пикселей при повороте. Ниже представлена реализация этого цикла на языке программирования Паскаль.

*x1:=x'-dx2; y1:=y'-dy2;*
*for i:=0 to n do begin*
*j:=0; x1:=x1+dx2;y1:=y1+dy2;*
*x2:=x1;y2:=y1;*
*S[i,j]:=S[x1,y1];*
*for j:=1 to m do begin*
*x2:=x2+dx1; y2:=y2+dy2;*
*S' [x2,y2]:=S[i,j];*
*end;*
*end;*

Предложенный метод позволяет из матрицы символа $S[i, j]$ размерности $n \times m$ получить матрицу $S'[i, j]$ большей размерности $]n \sin\varphi + m\cos\varphi[\times]n\cos\varphi + m\sin\varphi[$ (здесь $]x[$ – наименьшее целое число, большее $x$), в которой $n \cdot m$ пикселей имеют определенные цвета. То есть матрица $S'[i, j]$ содержит пиксели, значение которым не присвоено. Для того, чтобы этого не было, нами предлагается метод обратного преобразования, т.е. повернуть изображение $S'[i, j]$ на угол $-\varphi$.

Предложенный метод показывает, как совершить поворот изображения символа, представленного матрицей $n \times m$ пикселей за $2n(m+1)+18$ операций сложения, в то время как алгоритмы поворота, описанные в литературе [3–7], осуществляют поворот не менее, чем за $2nm$ операций умножения и $2nm$ операций сложения. Следовательно, предлагаемый алгоритм позволяет сэкономить $2nm$ операций умножения. С учетом того, что операция умножения требует больше затрат ресурсов, чем операция сложения (для целых чисел можно положить, что тратится в $\alpha = 4$), то экономится существенное количество времени – $\frac{\alpha}{\alpha+1} \cdot 100\% = 80\%$.

Количество операций в описанном алгоритме можно существенно сократить, если использовать в нем алгоритм Брезенхема [6, 7] для построения прямых.

**4. Экономичный метод быстрого поворота**

При этом методе из таблицы считываются не значения синуса и косинуса угла $\varphi$, а тангенса и котангенса, и полагается $d \times 1 : h1 \cdot tg(\varphi);\ dy1 : h2 \cdot ctg(\varphi)$. Для вычисленных значений $x'$ и $y'$ новые значения вычисляются по циклу:

*x1:=x'-dx2; y1:=y'-dy2;*
*for i:=0 to n do begin*
*j:=0; x:=x+dx1; y1:=y1+1;*
*if int{x}=int{x-dx1} then x1:=x1+1;*
*x2:=x1;y2:=y1;*
*S[i,j]:=S[x1,y1];*
*for j:=1 to m do begin*

*x2:=x2+1; y:=y+dy1;*
*if int{y2}=int{y2-dy1} then y2:=y2+1;*
*S' [x2,y2]:=S[i,j];*
*end;*
*end;*

Этот алгоритм выполняет меньшее количество операций за счет того, что здесь совершается лишь одно сложение вещественных чисел, а две другие операции сложения в теле цикла совершаются для целых чисел, которые выполняются в 4 раза быстрее (если принять вещественные числа 16-разрядными). То есть время выполнения уменьшится на 25% и станет 15% от времени выполнения алгоритмов [3–5].

Для избежания появления пустых пикселей, которые приводят к искажению символа, можно воспользоваться описанным способом обратного преобразования.

**5. Быстрый поворот векторного изображения**

Данный метод можно применить и для ускорения поворота контура символа, т.е. изображения символа, представленного в векторной форме.

Действительно, пусть контур символа представлен массивом $x[i], y[i] \quad i \in 0, m$. Его можно представить в кодировке Фримена, в которой каждой $i$-й точке массива сопоставлено число $a[i]$ от 0 до 7, показывающее направление обхода контура в точке $(x[i], y[i])$ (за 0 принимается направление вектора *Oy*) [8]. Таким образом, контур представляется в виде начальных точек $(x[0], y[0])$ и массива $m$ чисел от 0 до 7. Каждому из этих чисел сопоставляются свои $dx$ и $dy$, которые показывают, насколько изменились направления ориентации контура при повороте на данный угол в абсолютных координатах, и соответственно записывается функция, которая производит соответствующие изменения координат $x1$ и $y1$ – это *function_a[i],* а такая процедура *function_a[i]* состоит из двух операций сложения.

Теперь для поворота символа, представленного массивом $x[i], y[i], \ i \in 0,..,m$, нужно осуществить цикл по $i$

*for i:=0 to m do begin*
*function_a[i];*
*S' [i]:=S[i].*

При таком алгоритме возможно возникновение пробелов в новом контуре при повороте тех участков контура, которые записаны в виде нечетного числа в кодировке Фримена. Для устранения такой возможности предлагается использовать метод умножения на $\sqrt{2}$, т.е. те участки, которые записываются нечетными числами в кодировке Фримена представлять не в виде $k$ следующих подряд нечетных чисел, а в виде $[k\sqrt{2}]$ подряд следующих нечетных чисел. Соответственно для каждого из полученных $[k\sqrt{2}]$ пикселей осуществляется операция поворота на заданный угол, но

все коэффициенты преобразования умножаются на $\frac{1}{\sqrt{2}}$. Таким образом, для нечетных чисел $i$ выполняется процедура $\frac{1}{\sqrt{2}}$ *function_a[i].*

Сказанное выше позволяет сделать вывод, что для поворота символа произвольной формы достаточно осуществить поворот контура символа, а затем для заполнения внутренности выделенного контура совершить операцию поворота по вышеописанному алгоритму на угол, равный $-\varphi$.

В результате количество операций при повороте контура будет в $2\sqrt{2}$ больше количества цифр в кодировке Фримена, что все равно меньше, чем количество операций при повороте с помощью синусно-косинусных преобразований (1) – (2), а также с помощью известных методов [3–5].

Отметим, что в кодировке Фримена количество записей для границы связной области можно уменьшить (сжать). Действительно, в цепочке целых чисел от 0 до 7 с высокой вероятностью эти числа будут повторяться. Следовательно, можно сжать данную последовательность либо с помощью стандартных алгоритмов сжатия, например, LZW, либо выделяя все повторяющиеся числа. То есть для каждого числа $i \in 0..7$, которое идет подряд $k$ ($k > 2$) раз, запоминается пара чисел $(i,k)$. Тем самым вместо последовательности чисел $i,i,...,i$ в память компьютера записывается пара $(i,k)$ (вместе со скобками).

Такая кодировка позволяет сократить как время выполнения операций сдвига, так и поворота. Действительно, при таком сжатии эти операции выполняются для каждого одиночного (без скобок) числа точно так же, как в [3, 4]. Причем при возникновении скобок в случае преобразования сдвига можно не считывать новые точки сдвига, а достаточно повторить нужную операцию для первого числа в скобках количество раз, равное второму числу в скобках.

В случае же выполнения операции изменения масштаба достаточно для соответствующей скобки уменьшить (или увеличить) количество повторений, которое равно второму числу в скобках, на коэффициент изменения масштаба.

Оценим, насколько операций сократятся процедуры аффинного преобразования при таком методе сжатия векторного изображения.

При выполнении процедуры параллельного переноса количество операций останется таким же, поскольку будет осуществлено перекрашивание только тех пикселей изображения, которые изменили свой цвет при выполнении этого преобразования. При выполнении операции поворота количество арифметических операций также не уменьшится. Зато существенно уменьшится количество операций при преобразовании масштаба, поскольку вместо выполнения двух операций умножения для каждого пикселя можно осуществить одну операцию умножения.

Если $p$ – количество пикселей в контуре фигуры, а $q$ – количество участков, записанных одной цифрой в кодировке Фримена, то количество операций при параллельном переносе будет $p$ операций сложения, при повороте $-2p$ операций сложения, а при изменении масштаба – $q$ операций умножения. С учетом того, что $q < p$ и в среднем для изображений географических

цифровых карт его можно принять равным $q = \frac{p}{2}$ [9], применение указанного метода сжатия приводит к экономии времени на 75% при выполнении операции изменения масштаба, поскольку время выполнения уменьшается с $2p$ до $q$.

Кроме того, применение такого алгоритма сжатия для хранения изображения позволяет использовать не $p$, а $q$ единиц памяти, так как для каждого участка, являющегося отрезком, достаточно сохранить его длину в пикселях, что, в свою очередь, дает экономию памяти до 50%.

Таким образом, для поворота изображения символа, представленного в векторной форме, целесообразно использовать описанный алгоритм поворота совместно с предлагаемой процедурой сжатия.

**6. Выводы**

Предлагаемый быстрый экономичный метод поворота растрового изображения сложного символа содержит не более $2S$ операций сложения ($S$ – количество пикселей в матрице изображения символа), что на $2S$ операций умножения меньше по сравнению с известными решениями. Это позволяет сэкономить время преобразования до 85%.

Для изображений, представленных в векторном виде, предложен комплексный контурный метод поворота сложного символа, основанный на представлении контура с помощью кодировки Фримена. Причем для устранения возникающих пробелов в тех участках контура, которые записываются нечетными числами в кодировке Фримена, предложено представлять не в виде $k$ следующих подряд нечетных чисел, а в виде $[k\sqrt{2}]$ подряд следующих нечетных чисел с последующим выполнением процедуры $\frac{1}{\sqrt{2}}$ *function_a[i]*, что позволяет сохранить качество изображения.

Полученные методы могут быть использованы для создания программного обеспечения в комплексах ГК РВ повышения безопасности полетов, а также для совершенствования программ компьютерной анимации и других системах, производящих преобразования изображений в условиях лимита времени.

**СПИСОК ЛИТЕРАТУРЫ**


1. Алиев Т.М., Вигдоров Д.И., Кривошеев В.П. Системы отображения информации. – М.: Высшая школа, 1988. – 223 с.
2. Гилой В. Интерактивная машинная графика: структуры данных, алгоритмы, языки: Пер. с англ. – М.: Мир, 1981. – 384 с.
3. Васюхин М.И. Алгоритмические и программно-аппаратные методы и средства построения интерактивных геоинформационных комплексов оперативного взаимодействия: Дис… д-ра техн. наук: 05.13.13 / Институт кибернетики НАН Украины. – К., 2002. – 414 с.
4. Васюхин М.И., Зорич И.С. Отображение динамики единичного видеосимвола в реальном времени // УСиМ. – 1997. – № 1/3. – С. 87 – 92.
5. Смолий В.В. Применение конформных отображений в процессе геометрических преобразований изображений динамических объектов // Наукові праці Донецького державного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка. – Донецьк: ДонДТУ, 2000. – Вип. 15. – С. 150 – 155.
6. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения: Учебное пособие. – М.: Диалог-МИФИ, 1995. – 288 с.
7. Абраш М. Таинства программирования графики. – К.: ЕвроСИБ, 1996. – 384 с.
8. Бутаков Е.А., Островский В.И., Фадеев И.Л. Обработка изображений на ЭВМ. – М.: Радио и связь, 1987. – 240 с.
9. Васюхін М.І., Головко Б.Б., Бородін В.А. Оцінка ефективності застосування принципу квадротомічного дерева при формуванні БД ГІС для растрових зображень // Вісник геодезії та картографії. – 2001. – № 1. – С. 37 – 39.