УДК 620.179.15:004.421.2

**А. И. Закидальский, Е. А. Цыбульская**
Институт проблем регистрации информации НАН Украины
ул. Н. Шпака, 2, 03113 Киев, Украина

# Реализация свертки с помощью рекурсивных процедур для реконструкционного 3D-алгоритма

*Рассмотрен метод вычисления свертки, основанный на применении быстрого преобразования Фурье (БПФ). Приведены варианты программной реализации свертки. Показано, что использование БПФ позволяет повысить эффективность вычисления свертки и уменьшить время ее выполнения.*

***Ключевые слова:*** *3D-реконструкция, свертка, быстрое преобразование Фурье.*

В настоящее время в 3D-реконструкции томографических изображений широкое распространение получили алгоритмы, основанные на свертке и обратном проецировании [1]. Это связано с тем, что процесс реконструкции тогда можно проводить по мере поступления проекционных данных, не ожидая окончания сканирования, что значительно сокращает общее время реконструкции. Свертка занимает приблизительно 10 % от общего количества операций, тем не менее, когда при больших входных данных реконструкция выполняется часами, необходимо минимизировать все составные части алгоритма. Задача состояла в том, чтобы реализовать наиболее эффективный алгоритм вычисления свертки для минимизации времени, необходимого на реконструкцию.

Сверткой двух последовательностей $x(n_1)$ и $h(n_2)$ [2] длиной $N1$ и $N2$ соответственно является последовательность:

$$y(n) = \sum_{m=0}^{n} x(m)h(n-m)\,,\ n = N1 + N2 - 1. \qquad (1)$$

При таком вычислении свертки (согласно (1)) необходимо выполнить приблизительно $n^2$ операций.

Для уменьшения количества операций применяется метод быстрой свертки, состоящий в следующем:

1) вычисляются последовательности $X(n)$ и $H(n)$ путем выполнения БПФ последовательностей $x(n_1)$ и $h(n_2)$;

2) полученные $X(n)$ и $H(n)$ почленно перемножаются;

3) выполняется обратное БПФ (ОБПФ) произведения $X(n)\cdot H(n)$.

Тогда для вычисления свертки таким способом необходимо выполнить

$$\boldsymbol{n}\ \mathbf{log_2}\,\boldsymbol{n}\ (\text{БПФ от } x(n_1)) + \boldsymbol{n}\ \mathbf{log_2}\,\boldsymbol{n}\ (\text{БПФ от } h(n_2)) + \boldsymbol{n}(X(n)\cdot H(n)) +$$

$$+\ \boldsymbol{n}\ \mathbf{log_2}\,\boldsymbol{n}\ (\text{ОБПФ от } X(n)\cdot H(n)) = \mathbf{3}\ \boldsymbol{n}\ \mathbf{log_2}\,\boldsymbol{n} + \boldsymbol{n}\ \text{операций.}$$

Основой алгоритма вычисления свертки является алгоритм БПФ — оптимизированный по скорости способ вычисления дискретного преобразования Фурье (ДПФ).

Дискретное преобразование Фурье заключается в поиске по последовательности $\{x\} = x_0, x_1, \ldots, x_{n-1}$ другой последовательности $\{X\} = X_0, X_1,..., X_{N-1}$, элементы которой вычисляются по формуле:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-j2\pi kn}{N}}. \qquad (2)$$

Обратное дискретное преобразование Фурье заключается в поиске по последовательности $\{X\} = X_0, X_1,..., X_{N-1}$ другой последовательности $\{x\} = x_0, x_1,\ldots, x_{n-1}$, элементы которой вычисляются по формуле:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{j2\pi kn}{N}}. \qquad (3)$$

Основным свойством этих преобразований является тот факт, что из последовательности $\{x\}$ получается (при прямом преобразовании) последовательность $\{X\}$, а если потом применить к $\{X\}$ обратное преобразование, то снова получится исходная последовательность $\{x\}$.

Величина $e^{\frac{-j2\pi}{N}}$ (поворачивающий множитель) постоянна и не изменяется при вычислении $N$ членов последовательности $\{X\}$. Обозначим $W_N = e^{\frac{-j2\pi}{N}}$. Тогда (2) примет вид:

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{kn}. \qquad (4)$$

Главная идея вычисления БПФ [3] заключается в следующем.

1. Сумма (2) из $N$ слагаемых делится на две суммы по $N/2$ слагаемых и вычисляется по отдельности. Для вычисления каждой из подсумм, надо их тоже разделить на две, и т.д.

2. Когда в подсумме останутся 2 слагаемых, вычисляются два члена последовательности $\{X\}$.

Для разбиения последовательности можно применять либо «прореживание по времени» (когда в первую сумму попадают слагаемые с четными номерами, а во вторую — с нечетными), либо «прореживание по частоте» (когда в первую сумму попадают первые $N/2$ слагаемых, а во вторую — остальные). Оба варианта равноценны. В силу специфики алгоритма приходится использовать только $N$, являющиеся степенями 2, т.е. исходные последовательности дополняются нулями до длины, равной степени двойки [5].

Рассмотрим случай прореживания по времени.

Определим последовательности $\{x_{[e]}\}$ — четные элементы и $\{x_{[o]}\}$ — нечетные элементы через последовательность $\{x\}$ таким образом:

$$x_{[e]n} = x_{2n}, \tag{5}$$

$$x_{[o]n} = x_{2n+1}, \quad n = 0, 1, ..., N/2 - 1,$$

$\{X_{[e]}\}$ — БПФ от $\{x_{[e]n}\}$, $\{X_{[o]}\}$ — БПФ от $\{x_{[o]n}\}$.

$N$-точечное ДПФ последовательности $\{x\}$ можно записать как

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{[e]n} W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x_{[o]n} W_N^{(2n+1)k}. \tag{6}$$

С учетом того, что $W_N^2 = \left[e^{j(2\pi/N)}\right]^2 = e^{j[2\pi/(N/2)]} = W_{N/2}$, получим:

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_{[t]n} W_{N/2}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x_{[o]n} W_{N/2}^{nk} = X_{[e]k} + W_N^k X_{[o]k} \text{ при } 0 \le k \le N/2 - 1. \tag{7}$$

А так как $W_N^{k+N/2} = -W_N^k$, то, аналогично (7), получаем:

$$X_k = X_{[e]k} - W_N^k X_{[o]k} \text{ при } N/2 \le k \le N - 1. \tag{8}$$

Ниже приведен программный код на языке C++, реализующий вычисление БПФ и обратного БПФ заданных последовательностей с прореживанием по времени [4]. Коэффициенты $W_N^k$ вычислены заранее следующим образом:

$$W_N^k = e^{-j2\pi k/N} = \cos(2\pi k/N) - j \sin(2\pi k/N). \tag{9}$$

Функция fft_t получает такие входные данные:

*ind* — тип преобразования: прямое или обратное;

**rem, *imm* — массивы чисел с плавающей запятой, обеспечивающие работу с комплексными числами;

*NN* — длина входной последовательности (степень 2).

Вычисленный результат преобразования — последовательность комплексных чисел — замещает данные во входных массивах. При прямом БПФ порядок следования отсчетов во входном массиве — прямой, в результирующем — инверсный. При обратном БПФ — наоборот.

```
void fft_t( bool ind, float *rem, float *imm, int NN )
{  float re, im ;
   int i, j, k, n12, kn12, m;

   for( i = 2, n12 = 1; i <= NN; n12 = i, i += i ) {
      for( k = 0; k < n12; ++k ) {
          for( j = k; j < NN; j += i ) {
                    // индекс элемента второй подпоследовательности
              kn12 = j + n12;
                                        // X * W_N^k
              re = rem[kn12] * wnk_re[j] - imm[kn12] * wnk_im[j];
              im = imm[kn12] * wnk_re[j] + rem[kn12] * wnk_im[j];
                                          // для обратного БПФ
              if( ind == false ) im = -im;

              rem[kn12] = rem[j] - re;    // второй элемент
              imm[kn12] = imm[j] - im;
              rem[j] = rem[j] + re;        // первый элемент
              imm[j] = imm[j] + im;
              }
          }
      }
   if( ind == false )                       // для обратного БПФ
      for( i = 0; i < NN; rem[i] /= NN, imm[i++] /= NN );
}
```

В варианте алгоритма БПФ с прореживанием по частоте входная последовательность $\{x_n\}$ разбивается на две последовательности по $N/2$ отсчетов:

$$\{x_{[1]n}\} = \{x_n\},\ n = 0, 1, \ldots, N/2 - 1, \tag{10}$$

$$\{x_{[2]n}\} = \{x_{n+N/2}\},\ n = 0, 1, \ldots, N/2 - 1,$$

$\{X_{[1]}\}$, $\{X_{[2]}\}$ — их ДПФ.

При таком разбиении преобразование Фурье $\{x_n\}$ можно записать как

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x_n W_N^{nk} + \sum_{n=\frac{N}{2}}^{N-1} x_n W_N^{nk} = \sum_{n=0}^{\frac{N}{2}-1} x_{[1]n} W_N^{nk} + \sum_{n=0}^{\frac{N}{2}-1} x_{[2]n} W_N^{(N/2+n)k}. \tag{11}$$

Учитывая, что $W^{Nk/2} = e^{-j\pi k}$, получим:

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} \left[ x_{[1]n} + e^{-j\pi k} x_{[2]n} \right] W_N^{nk} \tag{12}$$

Запишем отдельно выражения для четных и нечетных отсчетов:

$$X_{2k} = \sum_{n=0}^{\frac{N}{2}-1} \left[ x_{[1]n} + x_{[2]n} \right] (W_N^2)^{nk} = \sum_{n=0}^{\frac{N}{2}-1} \left[ x_{[1]n} + x_{[2]n} \right] W_{N/2}^{nk} = X_{[1]k} + X_{[2]k}\,, \tag{13}$$

$$X_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1} \left[ x_{[1]n} - x_{[2]n} \right] W_N^{n(2k+1)} = \sum_{n=0}^{\frac{N}{2}-1} \left[ (x_{[1]n} - x_{[2]n}) W_N^n \right] W_{N/2}^{nk} = (X_{[1]k} - X_{[2]k}) W_N^k\,. \tag{14}$$

Алгоритм вычисления БПФ и обратного БПФ с прореживанием по частоте (используя выражения (13), (14)) реализован с помощью рекурсивной процедуры. Ниже приведен программный код данной процедуры на языке C++. Здесь также коэффициенты $W_N^k$ вычислены заранее согласно (9).

Функция fft_c получает такие входные данные:

*ind* — тип преобразования: прямое или обратное;

**rem*, **imm* — массивы чисел с плавающей запятой, обеспечивающие работу с комплексными числами;

*NN* — длина входной последовательности (степень 2);

*kw* — индекс коэффициента $W_N^k$.

Вычисленный результат преобразования — последовательность комплексных чисел — замещает данные во входных массивах. При прямом БПФ порядок следования отсчетов во входном массиве — прямой, в результирующем — инверсный. При обратном БПФ — наоборот.

```
void fft_c( bool ind, float *rem, float *imm, int NN, int kw )
{  float re, im, m;

   if( NN > 2 ) {
                              // первая половина отсчетов
       fft_c( ind, rem, imm, NN / 2, kw );
                              // вторая половина отсчетов
       fft_c( ind + NN / 2, imm + NN / 2, NN / 2, kw + NN / 2 );
       }
   else {                     // осталось два отсчета
       re = rem[0] - rem[1];
       im = imm[0] - imm[1];
       rem[0] = rem[0] + rem[1];    // первый элемент
```

```
        imm[0] = imm[0] + imm[1];
        if( !ind ) m = -Wnk_im[kw];        // для обратного БПФ
        else m = Wnk_im[kw];
        rem[1] = re * Wnk_re[kw] - im * m;      // второй элемент
        imm[1] = im * Wnk_re[kw] + re * m;
        }

   if( !ind )                 // для обратного БПФ
      for( i = 0; i < NN; rem[i] /= NN, imm[i++] /= NN );
}
```

Теперь покажем как реализован алгоритм быстрой свертки с использованием какой-либо из приведенных функций вычисления быстрого преобразования Фурье.

Функция conv получает такие входные данные:

**data*, **kern* — сворачиваемая последовательность и ядро свертки — массивы чисел с плавающей запятой;

*Nd*, *Nk* — длина входной последовательности и ядра (произвольные).

Вычисленный результат преобразования находится в массиве `svd_re`.

```
float *svd_re, *svd_im, *svk_re, *svk_im, *Wnk_re, *Wnk_im;
float PI = 3.1415926;

void conv ( float *data, float *kern, int Nd, int Nk )
{
   float v, tmp;
   int n, i, N2;

   n =  Nd + Nk -1;                  // длина свертки
   N2 = 2;
   while ( N2 < n ) N2 = N2 * 2;     // степень 2 для БПФ
                                     // коэффициенты Wnk
   wnk_re = new float [N2];
   wnk_im = new float [N2];
   v = 2 * PI / N2;
   for( i = 0; i < N2; ++i ) {
      wnk_re[i] = cos( v * i );
      wnk_im[i] = - sin( v * i );
      }
                              // сворачиваемая последовательность
   svd_re = new float [N2];
   svd_im = new float [N2];
   for( i = 0; i < N2; ++i ) {
      if( i < Nd ) svd_re[i] = data[i];
      else swd_re[i] = 0.0;
```

```
      svd_im[i] = 0.0;
      }
                              // ядро свертки
   svk_re = new float [N2];
   svk_im = new float [N2];
   for( i = 0; i < N2; ++i ) {
      if( i < Nd ) svk_re[i] = kern[i];
      else svk_re[i] = 0.0;
      svk_im[i] = 0.0;
      }
                              // БПФ последовательности
   fft_c( 0, svd_re, svd_im, N2, 0 );
                              // БПФ ядра
   fft_c( 0, svk_re, svk_im, N2, 0 );
                              // перемножение БПФ
   for( i = 0; i < N2; ++i ) {
      tmp = svd_re[i] * svk_re[i] - svd_im[i] * swk_im[i];
      svd_im[i] = svd_re[i] * svk_im[i] + svd_im[i] * swk_re[i];
      svd_re[i] = tmp;
      }
                              // вычисление обратного БПФ
   fft_c( 1, svd_re, svd_im, N2, 0 );
}
```

Оба варианта вычисления свертки (с рекурсивным и нерекурсивным вычислением БПФ) проверялось на входных последовательностях длиной 512 ($2^9$) и 1024 ($2^{10}$) отсчета на компьютере P-IV 1,5 гГц. Временные характеристики работы приведены в таблице.

| **Входные данные** | **Время работы** | | |
|---|---|---|---|
| | Прямая свертка | Нерекурсивное БПФ | Рекурсивное БПФ |
| 200 × 512 | 2 мин 55 сек | 43 сек | 41 сек |
| 200 × 1024 | 8 мин 18 сек | 1 мин 56 сек | 1 мин 49 сек |

Полученные результаты показывают, что вычисление свертки с использованием БПФ дает выигрыш во времени в 3–4 раза, причем использование рекурсивной процедуры вычисления БПФ более эффективно за счет более простой индексации входной последовательности. Использование этой процедуры позволит сократить время выполнения свертки в процессе реконструкции.

1. *Терновой К.С., Синьков М.В., Закидальский А.И., Яник А.Ф. и др.* Введение в современную томографию. — К.: Наук. думка, 1983. — 345 с.

2. *Рабинер Л., Гоулд Б.* Теория и применение цифровой обработки сигналов. — М.: Мир, 1978. — 848 с.

3. *Cooley J.W. and Tukey J.W.* An algorithm for the machine calculation of complex Fourier series // Math. Comput. — 1965. — Vol. 19, N 90. — P. 297–301.

4. *Войнаровский Мирослав.* Программирование. Быстрое преобразование Фурье. — 2002. — http://psi-logic.narod.ru/fft/fft1.htm

5. *Просеков О.В.* Разработка алгоритма БПФ для нетрадиционного числа точек // Сб. докладов I научно-технической конф. молодых специалистов ЦНИИ «Морфизприбор» — Санкт-Петербург (Россия). — 2003, 22–25 апреля. — С. 116–121.

Поступила в редакцию 03.06.2005