КОМП'ЮТЕРНА ГРАФІКА.
/Обробка та стиск зображень./

Анатолій О.Лигун,
Олександр О.Шумейко

Дніпродзержинськ

2007

Зміст

Вступ

Комп'ютерна графіка є популярною областю computer science, що забезпечує її швидкий розвиток. Це обумовлено, перш за все, тими можливостями, які надають сучасні обчислювальні засоби. Все, що пов'язане з графічними образами, завжди викликало у людини підвищений інтерес. Не випадково однією з найпопулярніших сфер комп'ютерної графіки є створення візуальних ефектів і програмування ігор.

Інтерес розробників до програмування візуальних ефектів підштовхнув до досліджень в цій сфері і, як наслідок, до появи безлічі публікацій. Нами задумано декілька книг, матеріал яких повинен охопити ті розділи комп'ютерної графіки, які мало відображені в російсько- та українсько-мовних виданнях. Перша з них [27] присвячена задачам векторного опису кривих. Наступна книга, яка планується до видання, присвячена рекурсивним методам, зокрема методам subdivision і wavelets.

Даний посібник несе на собі функції підготовки читача до роботи з растровою графікою. У посібнику відображені теми, пов'язані з обробкою зображень і задачею стиску даних.

Розділ 1

Обробка зображень.

1.1 Простори кольорів.

Колір є результатом сприйняття світла в сітківці ока в спектрі, що має довжину хвилі від 400 nm до 700 nm. Сітківка людського ока має три типи рецепторів, кожний з яких відповідає за сприйняття світла визначеної частини спектру. Четвертий тип рецепторів, присутніх в сітківці – палички, які ефективні тільки на надзвичайно низьких рівнях світла (образно кажучи – нічний зір). Хоча для зору вони достатньо важливі, ніякої ролі для сприйняття кольору вони не виконують. Оскільки для сприйняття кольору існують рівно три типи рецепторів, то саме три колірні компоненти необхідні для опису кольору.

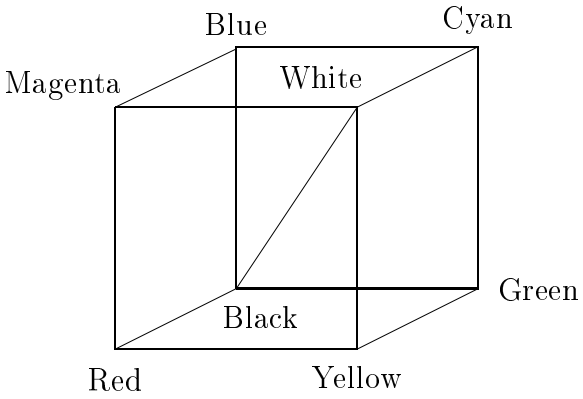
Зі всіх існуючих колірних моделей найбільш популярні наступні представлення кольорів (простори кольорів)

- RGB – модель змішування кольорів (базова комп'ютерна модель),
- CMYK – модель віднімання кольорів (використовується при кольоровому друці),
- YIQ, YUV, YCrCb (використовуються у відео - системах).

Проте жодна з цих моделей не дає інтуїтивний опис через тональність (hue), насиченість (saturation) і яскравість (brightness). Для такого роду опису кольорів використовуються інші колірні моделі, такі як HSI і HSV. Але всі ці системи так чи інакше використовують RGB інформацію, одержану з камери або зі сканера.

1.1.1 Колірний простір RGB

Найпоширенішою і часто використовуваною в комп'ютерній графіці моделлю є модель змішування трьох базових кольорів – червоного, зеленого та синього. Змішування базових кольорів в різних пропорціях дозволяє одержати всю колірну палітру. Для ілюстрації RGB моделі зручно використовувати діаграму у вигляді куба



У кубі кольорів моделі RGB початок координат відповідає чорному кольору, а на діагоналі, що йде від початку координат всі базові кольори змішуються в однаковій пропорції, тому діагональ містить всі відтінки сірого від чорного до білого.

Проте, у зв'язку з тим, що у відтінках сірого світлі тони володіють вищою інтенсивністю, ніж темні, для перетворення кольорів з простору RGB у відтінки сірого використовується формула

$$R_g = G_g = B_g = \frac{1}{256}(77R + 150G + 29B + 128).$$

Модель кольорів CMYK (Cyan -блакитний, Magenta - пурпуровий, Yellow жовтий) реалізується на пристроях, які використовують принцип поглинання кольорів. Оскільки чорний виходить при поглинанні всіх, що важко реалізувати на практиці, то в цій моделі використовується ще чорний колір -blacK. Якщо всі компоненти кольорів кодуються числами від 0 до 255, то має місце таблиця перекодування

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 255 - R \\ 255 - G \\ 255 - B \end{pmatrix}.$$

1.1.2 Простори кольорів YUV, YIQ, YCrCb.

Популярність RGB моделі пояснюється, перш за все, тим, що відображення кольору на екрані монітора реалізується саме змішуванням базових кольорів. Проте великим недоліком цієї системи є рівноправність всіх колірних компонентів. Цей чинник стимулював появу інших систем перенесення кольорів, в яких основне інформаційне навантаження несе одна колірна компонента – люмінесцентна складова (яскравість зображення). Для доповнення люмінесцентної складової до оригінального сигналу існують дві колірні компоненти. Структура цих компонентів визначає ту або іншу систему перенесення кольорів – YUV, YIQ, YCrCb.

У 1953 р. Національний комітет з телевізійних систем (NTSC – National Television System Committee) прийняв як стандарт колірну систему YIQ, засновану на моделі МКО XYZ. Через обмеження на ширину смуги пропускання, яскравість визначається однією координатою Y. Сигнал Y займає основну частину смуги частот (0-4 Мгц),

причому в ньому пропорції червоного, зеленого і синього основних кольорів NTSC вибрані так, що він відповідає кривій спектральної чутливості ока. У сигналі Y міститься інформація про яскравість, тому в чорно-білому телебаченні використовується тільки ця координата. Для того, щоб передавати колір, тобто тон і насиченість, за допомогою вужчої смуги частот, враховуються деякі особливості зорового сприйняття. Зокрема, чим менше предмет, тим гірше розрізняється його колір, а об'єкти, менші певного розміру, здаються чорно-білими. Якщо ж об'єкт менше за деяку мінімальну межу, то його колір взагалі не сприймається. У системі YIQ інформація про тон і насиченість кольору представляється за допомогою лінійних комбінацій різниць червоного, зеленого і синього кольорів і значення Y. Координата кольору I (синфазний сигнал) відповідає кольорам від помаранчового до зеленого, тобто "теплим"тонам, Q (інтегрований сигнал) - від зеленого до пурпурового, тобто всім іншим. Координата I займає смугу частот приблизно 1.5 Мгц, а Q - тільки 0.6МГц.

Для зв'язку між RGB і YIQ використовується рівність

$$\begin{cases} Y & = & 0.299R + 0.587G + 0.114B, \\ I & = Red - Cyan = & 0.596R - 0.275G - 0.3216B, \\ Q & = Magenta - Green = & 0.212R - 0.523G + 0.311B, \end{cases}$$

і, відповідно,

$$\begin{cases} R & = Y + 0.956I + 0.621Q, \\ G & = Y - 0.272I - 0.647Q, \\ B & = Y - 1.107I + 1.704Q. \end{cases}$$

Помітимо, що для традиційного однобайтового діапазону зміни базових кольорів RGB, область значень Y від 0 до 255, I змінюється в проміжку від 0 до ± 152, а V в проміжку від 0 до ± 134.

Колірний простір YUV використовується для передачі кольорового зображення в телевізійних системах PAL (Phase Alternation Line) і SECAM (Sequentiel Couleur Avec Mémoire or Sequential Color with Memory).

Співвідношення, що дозволяє зв'язати компоненти YUV з базовими кольорами RGB, виглядає таким чином

$$\begin{cases} Y & = 0.299R + 0.587G + 0.114B, \\ U & = -0.147R - 0.289G + 0.436B \\ & = 0.492(B - Y), \\ V & = 0.615R - 0.515G - 0.100B \\ & = 0.877(R - Y) \end{cases}$$

і, відповідно,

$$\begin{cases} R & = Y + 1.140V, \\ G & = Y - 0.395U - 0.581V, \\ B & = Y + 2.032U. \end{cases}$$

Компоненти I,Q пов'язані з U,V таким чином

$$\begin{bmatrix} I \\ Q \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \cos 33^0 & \sin 33^0 \\ -\sin 33^0 & \cos 33^0 \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix}$$

Зазначимо, що, якщо базові кольори RGB змінюються в діапазоні від 0 до 255, то область значень Y також від 0 до 255, а U лежить в діапазоні від 0 до ± 112, а V від 0 до ± 157.

Колірний простір YCrCb (люмінесцентна складова, хроматичний червоний, хроматичний синій) є найбільш поширеним серед комп'ютерних відео - стандартів. Саме цей колірний простір використовується в таких популярних форматах як JPEG, MPEG, Kodak's Photo YCC. Простір YCrCb є масштабованим в один байт (по кожній компоненті) колірний простір YUV.

Існує декілька модифікацій цього колірного простору. У YCrCb-SDTV колірна компонента Y займає 8 біт і змінюється в діапазоні від 16 до 235, область зміни Cr і Cb від 16 до 240.

$$\begin{cases} Y = & 0.299R + 0.587G + 0.114B, \\ Cb = & -0.172R - 0.339G + 0.511B + 128, \\ Cr = & 0.511R - 0.428G - 0.083B + 128, \end{cases}$$

і, відповідно,

$$\begin{cases} R & = Y + 1.371(Cr - 128), \\ G & = Y - 0.698(Cr - 128) \\ & \quad -0.336(Cb - 128), \\ B & = Y + 1.732(Cb - 128). \end{cases}$$

Як видно, під кожне значення колірної компоненти відводиться по вісім біт, то одержана система перенесення кольорів не повністю використовує виділені ресурси. Для того, щоб мати нагоду повністю використовувати байт, виділений під кожну колірну компоненту, одержані метод опису кольорів можна модернізувати таким чином

$$\begin{cases} Y = & 0.257R + 0.504G + 0.098B + 16, \\ Cb = & -0.148R - 0.291G + 0.439B + 128, \\ Cr = & 0.439R - 0.368G - 0.071B + 128, \end{cases}$$

і, відповідно,

$$\begin{cases} R & = 1.164(Y - 16) + 1.596(Cr - 128), \\ G & = 1.164(Y - 16) - 0.813(Cr - 128) \\ & \quad -0.391(Cb - 128), \\ B & = 1.164(Y - 16) + 2.018(Cb - 128). \end{cases}$$

У YCrCb-HDTV колірна компонента Y займає 8 біт і змінюється в діапазоні від 16 до 235, область зміни Cr і Cb від 16 до 240.

$$\begin{cases} Y = & 0.213R + 0.715G + 0.072B, \\ Cb = & -0.117R - 0.394G + 0.511B + 128, \\ Cr = & 0.511R - 0.426G - 0.047B + 128, \end{cases}$$

$$\begin{cases} R & = Y + 1.540(Cr - 128), \\ G & = Y - 0.459(Cr - 128) \\ & \quad -0.183(Cb - 128), \\ B & = Y + 1.816(Cb - 128). \end{cases}$$

Комп'ютерна реалізація цього простору кольорів наступна

$$\begin{cases} Y = & 0.183R + 0.614G + 0.062B + 16, \\ Cb = & -0.101R - 0.338G + 0.439B + 128, \\ Cr = & 0.439R - 0.399G - 0.040B + 128, \end{cases}$$

$$\begin{cases} R & = 1.164(Y - 16) + 1.793(Cr - 128), \\ G & = 1.164(Y - 16) - 0.534(Cr - 128) \\ & \quad -0.213(Cb - 128), \\ B & = 1.164(Y - 16) + 2.115(Cb - 128). \end{cases}$$

У стандарті JPEG2000 використовується наступна схема

$$\begin{cases} Y & = 0.299R + 0.587G + 0.144B, \\ Cb & = -0.16875R - 0.33126G + 0.5B, \\ Cr & = 0.5R - 0.41869G - 0.08131B, \end{cases}$$

$$\begin{cases} R = & Y + 1.402Cr, \\ G = & Y - 0.34413Cr - 0.71414Cb, \\ B = & Y + 1.772Cb. \end{cases}$$

Ще однією модифікацією є розробка Eeastman Kodak Company – колірний простір PhotoYCC. Для цього колірного простору люмінесцентна складова і хроматичні сигнали знаходяться таким чином

$$\begin{cases} Y = & 0.213R + 0.419G + 0.081B, \\ C1 = & -0.131R - 0.256G + 0.387B + 156, \\ C2 = & 0.373R - 0.312G - 0.061B + 137, \end{cases}$$

$$\begin{cases} R & = 0.981Y + 1.315(C1 - 137), \\ G & = 0.981Y - 0.311(C1 - 156) \\ & \quad -0.699(C2 - 137), \\ B & = 0.981Y + 1.601(C2 - 156). \end{cases}$$

Для того, щоб реалізувати швидке відновлення кольору, використовуються формули, що вживають розподіл тільки на числа рівні ступені числа два, тобто проводиться розподіл бітовим зрушенням. Приведемо такі формули:

$$\begin{cases} Y = & \frac{1}{23}\left(7R + 14G + 2B\right), \\ Cr = & \frac{4}{69}\left(8R - 7G - B\right), \\ Cb = & -\frac{4}{23}\left(R + 2G - 3B\right), \end{cases}$$

$$\begin{cases} R & = Y + \frac{3}{2}Cr, \\ G & = Y - \frac{1}{4}(3Cr + Cb), \\ B & = Y + \frac{7}{4}Cb. \end{cases}$$

На закінчення цього параграфа приведемо одну математичну конструкцію колірного простору YUV.

У якості Y візьмемо якнайкраще наближення трьох компонентів RGB однією компонентою. Неважко бачити, що в цьому випадку

$$Y = \frac{1}{3}(R + G + B),$$

тобто Y приймає значення, що лежать на головній діагоналі куба кольорів RGB.

Похибка відновлення кожного кольору цієї складової буде дорівнювати

$$\Delta R = R - Y = \frac{2}{3}R - \frac{1}{3}G - \frac{1}{3}B;$$

$$\Delta G = G - Y = \frac{2}{3}G - \frac{1}{3}R - \frac{1}{3}B;$$

$$\Delta B = B - Y = \frac{2}{3}B - \frac{1}{3}G - \frac{1}{3}R.$$

Далі знайдемо якнайкраще наближення пари $\Delta R, \Delta G$ однією компонентою U:

$$U = \frac{1}{2}(\Delta R + \Delta G) = \frac{1}{6}R + \frac{1}{6}G - \frac{1}{3}B.$$

Похибка відновлення компоненти G буде рівна

$$\Delta^2 G = \Delta G - U = -\frac{1}{2}R + \frac{1}{2}G.$$

Якнайкраще наближення пари $\Delta B, \Delta^2 G$ компонентою V буде таким

$$V = \frac{1}{2}(\Delta^2 G + \Delta B) = -\frac{5}{12}R + \frac{1}{12}G + \frac{1}{3}B.$$

В результаті одержуємо новий колірний простір YUV, що визначається рівністями

$$Y = \frac{1}{3}(R + G + B),$$

$$U = \frac{1}{6}R + \frac{1}{6}G - \frac{1}{3}B,$$

$$V = -\frac{5}{12}R + \frac{1}{12}G + \frac{1}{3}B.$$

Вирішуючи зворотну задачу, одержуємо

$$R = -U - 2V;$$

$$G = +3U + 2V;$$

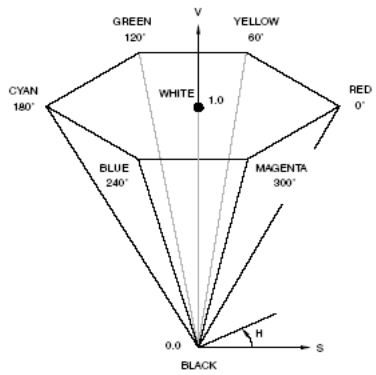
$$B = Y - 2U.$$

1.1.3 Простори кольорів інтуїтивного сприйняття.

Розглянуті простори кольорів зручні для тієї або іншої апаратної реалізації, проте описувати суб'єктивне сприйняття кольору людьми в цих системах незручно.

Наприклад, модель HSV базується на інтуїтивно близьких людині поняттях колірного тону H (hue) - довжина хвилі домінуючої компоненти в спектрі світлового потоку, насиченості S (saturation) - частина білого кольору (тобто частина спектру, в якому присутні всі частоти видимого діапазону), інтенсивності V (value) - загальна кількість світлового потоку.

Якщо колірний куб RGB спроектувати на площину уподовж чорно-білої діагоналі, виходить шестикутник з основними і додатковими кольорами у вершинах. При зниженні насиченості або чистоти основних кольорів відповідна шестикутна проекція також буде менше. Якщо проекції куба RGB зібрати уздовж головної діагоналі, що представляє собою кількість світла, або світлову складову кольору від чорного (0) до білого (1), то вийде об'ємний шестигранний конус моделі HSV.



Інтенсивність уздовж його осі зростає від 0 у вершині до 1 на верхній грані, де вона максимальна для всіх кольорів. Насиченість визначається відстанню від осі, а тон - кутом (від 0^0 до 360^0), що відлічується від червоного кольору. Для того, щоб на малюнку червоний був на початку відліку, проекція колірного куба була повернена на 120^0 проти годинникової стрілки. Значення H вимірюється в градусах від 0 до 360 у порядку стадії нагрівання - червоний, жовтий, зелений, блакитний, синій, фіолетовий.

Насиченість міняється від 0 на осі до 1 на межі шестикутника. Відзначимо, що насиченість залежить від відстані від осі до межі для кожного V. При S=1 кольори або їх доповнення повністю насичені. Ненульова лінійна комбінація трьох основних кольорів не може бути повністю насичена. Якщо $S = 0$, то тон H невизначений, тобто на центральній осі знаходяться ахроматичні, сірі кольори. Модель HSV відповідає тому, як складають кольори художники. Чистим пігментам відповідають значення $V = 1$, $S = 1$, розбігам - кольори зі збільшеним змістом білого, тобто з меншим S, відтінкам - кольори із зменшеним V, які виходять при додаванні чорного. Тон змінюється при зменшенні як V, так і S.

Перетворення колірного простору HSV в RGB виконується безпосередньо за допомогою геометричних співвідношень між колірним шестигранним конусом і кубом.

Алгоритм перетворення HSV в RGB.

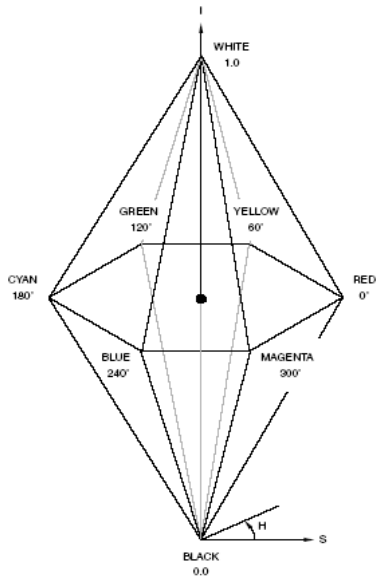
```
 H - колірний тон (0-360°),
 0° - червоний
 S - насиченість (0-1)
 V- освітленість (0-1)
 RGB - червоний, зелений, синій;
 основні кольори (0-1)
 Floor - функція, що виділяє цілу частину числа
 Перевірка ахроматичного випадку
 if S = 0 then
      if H = Невизначеність then
         R = V
         G = V
         У = V
      else
         якщо H визначене, то помилка
     end if
else
     хроматичний випадок
     if H = 360 then
        H = 0
     else
        H = H/60
     end if
     I = Floor (H)
     F = H - I
     M = V * (1 - S)
     N = V*(1-S*F)
     K = V * (1 - S * (1 - F))
     (R, G, B) = (V, K, M)  що значить R = V, G = K, B = M і т. д.
     if I = 1 then (R, G, B) = (V, K, M)
     if I = 2 then (R, G, B) = (N, V, M)
     if I = 3 then (R, G, B) = (M, V, K)
     if I = 4 then (R, G, B) = (M, N, V)
     if I = 5 then (R, G, B) = (K, M, V)
     if I = 6 then (R, G, B) = (V, M, N)
end if finish }
```

Алгоритм перетворення RGB в HSV.

```
RGB - червоний, зелений, синій;
основні кольори (0-1)
H - колірний тон (0-360°), 0° - червоний
S -насиченність (0-1)
V - освітленість (0-1)
```

```
Max - функція визначення максимуму
Min - функція визначення мінімуму визначення освітленості
V = Max (R, G, B) визначення насиченості
Temp = Min (R, G, B) if V = 0 then
   S = 0
else
   S = (V - Temp)/V
end if визначення колірного тону if S = 0 then
   H = Невизначеність
else
   Cr = (V - R)/(V - Temp)
   Cg = (V - G)/(V - Temp)
   Cb = (V - B)/(V - Temp)
   колір між жовтим і пурпуровим
   if R = V then H = Cb - Cg
   колір між блакитним і жовтим
   If G = V then H = 2 + Cr - Cb
   колір між пурпуровим і блакитним
   if B = V then H = 4 + Cg - Cr
   переклад в градуси
   H = 60 * H
   приведення до позитивних величин
   If H < 0 then H = H + 360
end if finish.
```

Узагальненням моделі HSV є колірний многогранник HSI, який відрізняється від HSV обчисленням компоненти інтенсивності кольору I (intensity).



Іншим узагальненням колірної моделі HSV є опис колірного простору не пірамідою, а конусом.

1.2 Інтерполяція і децимація.

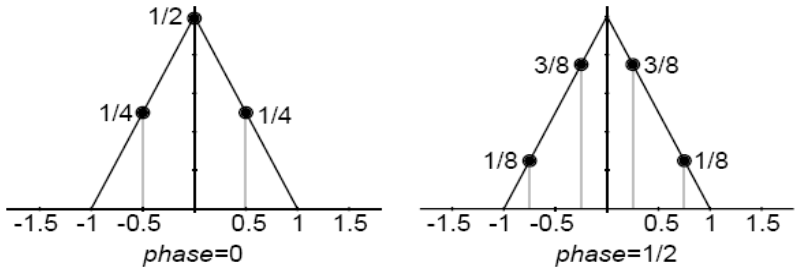
Через дискретність відображення сигналів (образів), вся графічна інформація розглядається як дискретна. Цей факт приводить до певних незручностей при масштабуванні зображень. При зменшенні масштабу (накат на образ, збільшення геометричного розміру зображення, підвищення глибини перенесення кольорів і ін.) кількість інформації, що відображається, збільшується, а при збільшенні масштабу (зменшенні геометричного розміру зображення, пониження глибини перенесення кольорів і ін.) її кількість зменшується. Процес збільшення точок, які відображаються, називається інтерполяцією, зменшення– децимацією (проріджування)(від латинського decimating - знищення кожного десятого полоненого воїна переможеної римлянами армії).

1.2.1 Методи інтерполяції і децимації одновимірних даних.

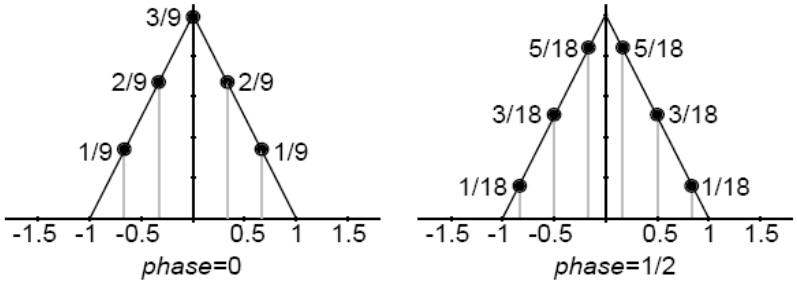
При реалізації алгоритмів інтерполяції і децимації можливо, що нове значення лежить в точці, в якій раніше було старе значення, в цьому випадку говорять, що фазове зрушення перетворення дорівнює нулю. Якщо координати нової точки лежать у середині між старими, то фазове зрушення рівне 1/2 і т.ін.

Найчастіше для побудови фільтрів використовуються різні інтерполяційні формули, як правило на основі інтерполяційних многочленів. Ми розглянемо найпростіший випадок алгебраїчної інтерполяції – лінійної. В цьому випадку функція, яка породжує фільтр, буде B-сплайном першого порядку.

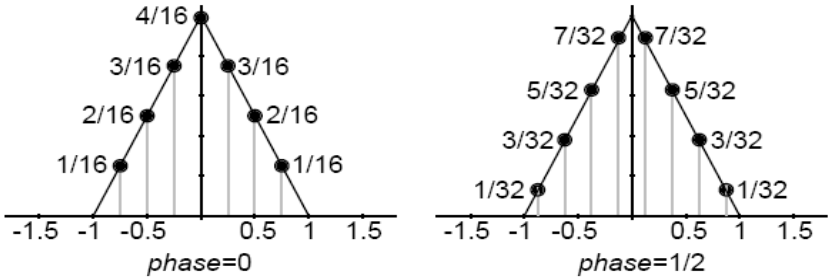
Фільтр для децимації даних в два рази може бути реалізований таким чином:



децимація в три рази:



децимація в чотири рази:



Для цього випадку, інтерполяція нових значень виходить зняттям цих значень з інтерполяційної ламаної.

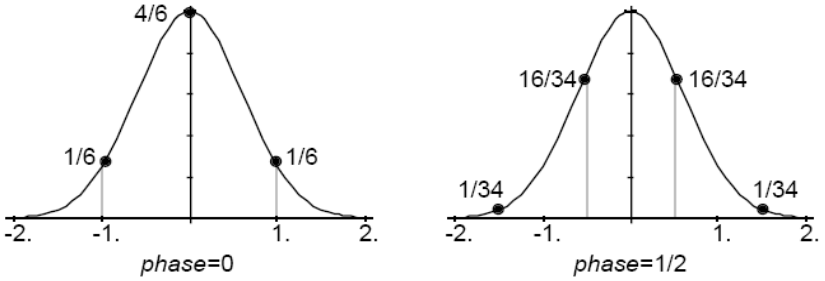
Однією з найчастіше використовуваних функцій для подібного роду задач є функція Гауса

$$y = e^{-a^2 x^2}.$$

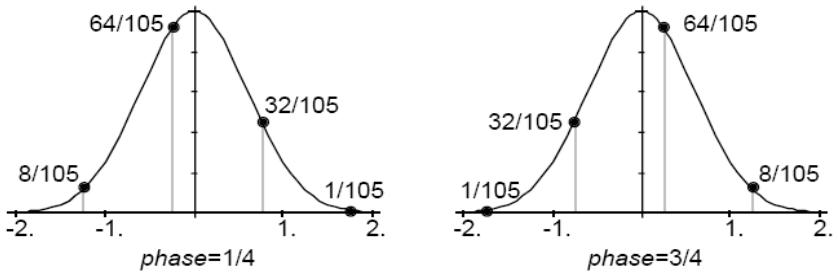
На наступних прикладах розглянемо функцію Гауса вигляду

$$y = 2^{-\frac{1}{2}x^2}.$$

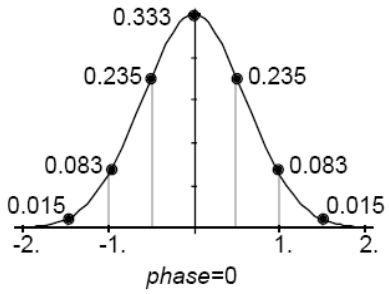
Для інтерполяції в два рази, слід використовувати наступну маску

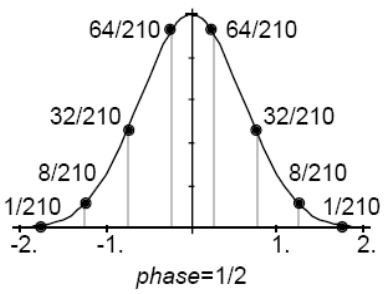


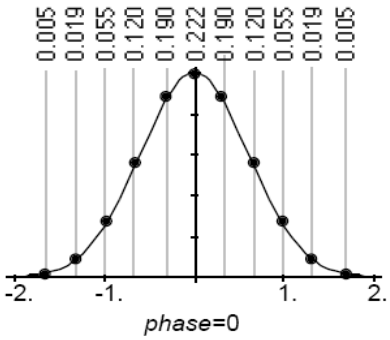
Інтерполяція в три рази виходить таким чином

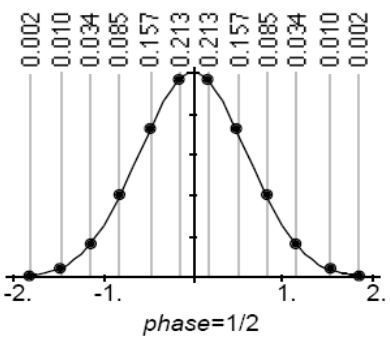


Як приклад приведемо маску децимації в два і три рази:

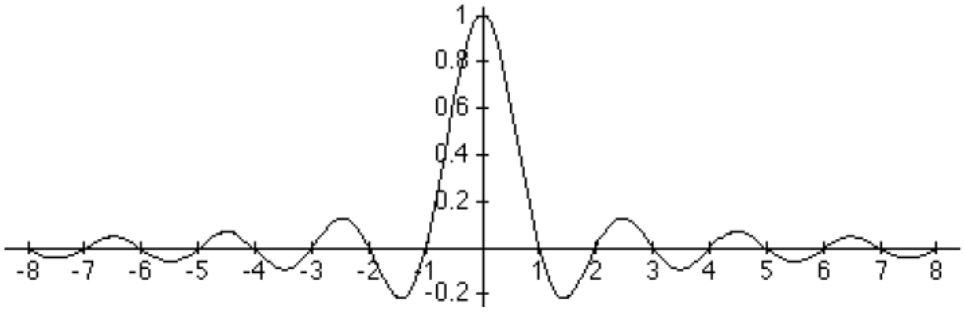








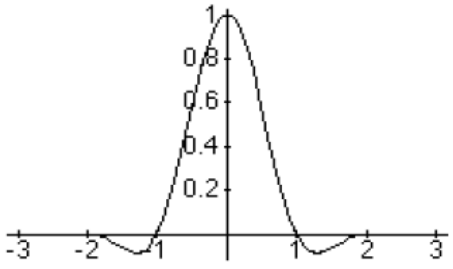
Ідеальною функцією для побудови низькочастотного фільтру є функція $\mathrm{sinc}(x)$



.

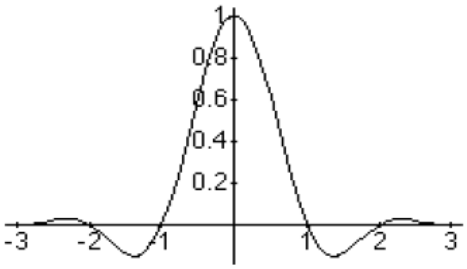
Внаслідок того, що значення цієї функції хоча і швидко затухають, але її носій не обмежений, то для побудови фільтрів на основі цієї функції беруть добуток її на деяку віконну функцію. Результатом будуть, наприклад, наступні функції

$$\text{Lanczos2(x)} = \begin{cases} \frac{\sin \pi x}{\pi x} \ \frac{\sin \pi \frac{x}{2}}{\pi \frac{x}{2}}, & |x| < 2 \\ 0, & |x| \geq 2, \end{cases}$$

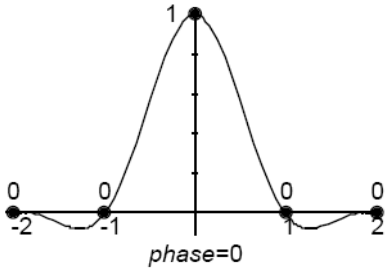


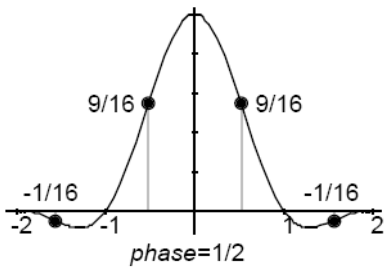
і

$$\text{Lanczos3(x)} = \begin{cases} \frac{\sin \pi x}{\pi x} \ \frac{\sin \pi \frac{x}{3}}{\pi \frac{x}{3}}, & |x| < 3 \\ 0, & |x| \geq 3. \end{cases}$$

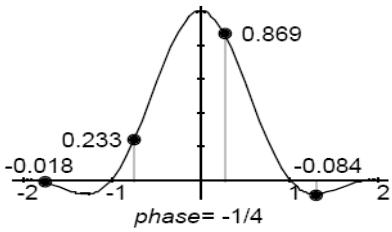


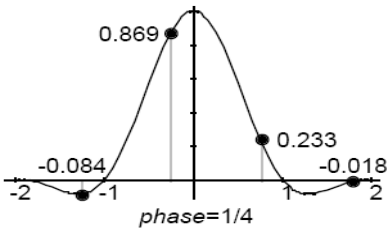
Приведемо фільтри інтерполяції в два рази, одержані на основі функції Lanczos2



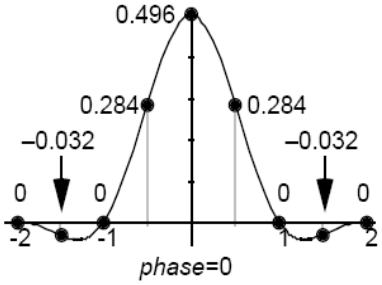


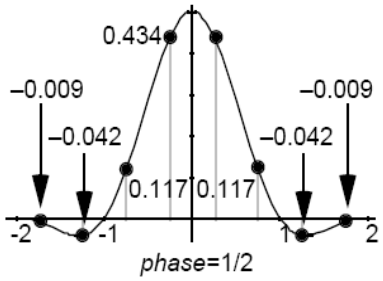
Помітимо, що при фазовому зрушенні рівним нулю, береться тільки одне значення.

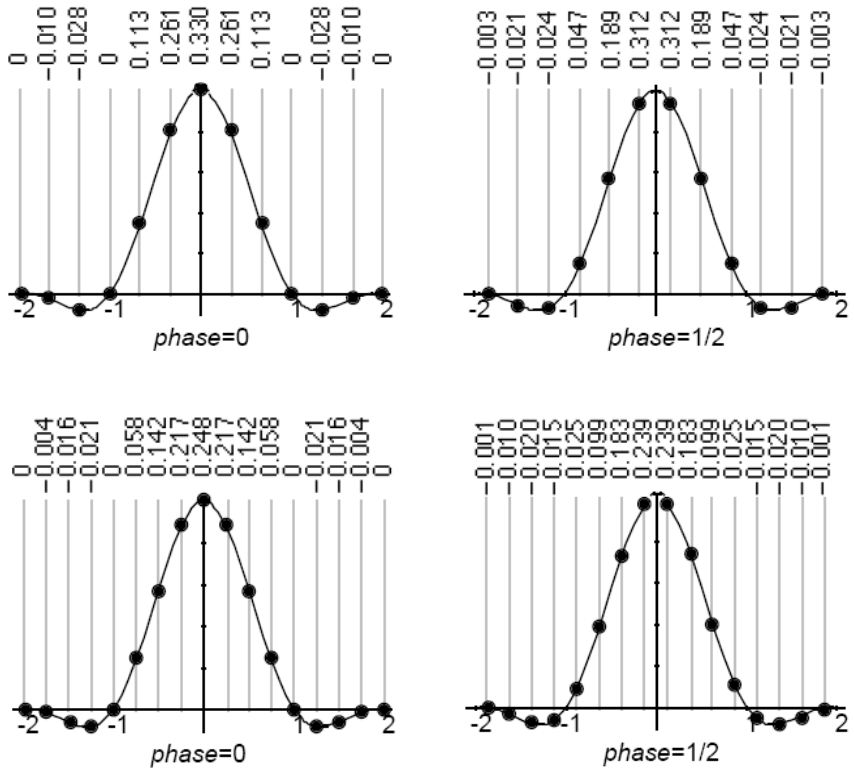




Для децимації в два, три і чотири рази маємо наступні фільтри







1.2.2 Метод бінірного зменшення для двовимірних даних.

В зв'язку з великим обсягом обчислень при візуалізації відео-файлів, як правило, фільми зберігаються у форматі cif, тобто з розміром кадру 320x240px, а при відтворенні відео, зображення розтягується. Таким чином, є актуальним розробка методів геометричного зменшення зображення з метою наступного збільшення його таким чином, щоб зменшити похибку відновлення.

Розглянемо одну конструкцію збільшення розміру, яка є оптимальною для заданого методу відновлення на фіксованій множині фільтрів. На першому кроці розглянемо відновлення функцій, неперервно диференційованих (до шостого порядку включно) на всій дійсній площині. Також будемо вважати, що задана деяка фіксована функція $\varphi(x, y)$ така, що $\varphi * Const(x, y) = Const$, центральні моменти якої

$$m_{\nu,\mu} = \int\int_{\mathbb{R}^2} x^\nu y^\mu \varphi(x, y)dxdy = \int\int_{\mathbb{D}} x^\nu y^\mu \varphi(x, y)dxdy$$

задовольняють умовам

$$m_{\nu,\mu} = m_{\mu,\nu}, m_{2\nu+1,\mu} = m_{\nu,2\mu+1} = 0,$$

а також, для відновлення константи необхідне виконання умови $m_{0,0} = 1$.

Будемо вважати, що інформація про функцію f задається множиною функціоналів

$$c_{2i,2j} = \int\int_{\mathbb{R}^2} f(x + 2ih, y + 2jh)\varphi(x, y)dxdy,$$

де h – крок решітки та $i, j \in \mathbb{Z}$.

Для відновлення функції f у вузлах решітки (ih, jh) побудуємо наступний алгоритм (типу метода Рунге-Кутта). На першому кроці побудуємо реконструкцію функції f у вузлах $(2ih, 2jh)$

$$\widehat{f}_{2i,2j} = \alpha_0 c_{2i,2j} + \frac{\alpha_1}{4}\left(c_{2i-2,2j} + c_{2i+2,2j} + c_{2i,2j-2} + c_{2i,2j+2}\right) +$$
$$+\frac{\alpha_2}{4}\left(c_{2i-2,2j-2} + c_{2i+2,2j-2} + c_{2i-2,2j-2} + c_{2i+2,2j+2}\right) +$$
$$+\frac{\alpha_3}{4}\left(c_{2i-4,2j} + c_{2i+4,2j} + c_{2i,2j-4} + c_{2i,2j+4}\right).$$

На наступному кроці обчислемо реконструкцію функції f у вузлах $((2i+1)h, (2j+1)h)$ з використанням отриманих значень $\widehat{f}_{2i,2j}$ за правилом

$$\widehat{f}_{2i+1,2j+1} = \frac{\beta_0}{4}\left(c_{2i,2j} + c_{2i+2,2j} + c_{2i,2j+2} + c_{2i+1,2j+2}\right) +$$
$$+\frac{\beta_1}{4}\left(\widehat{f}_{2i,2j} + \widehat{f}_{2i+2,2j} + \widehat{f}_{2i,2j+2} + \widehat{f}_{2i+1,2j+2}\right).$$

Нарешті, з урахуванням всіх отриманих значень, знайдемо

$$\widehat{f}_{2i+1,2j} = \frac{\gamma_0}{2}\left(c_{2i,2j} + c_{2i+2,2j}\right) +$$
$$+\frac{\gamma_1}{2}\left(\widehat{f}_{2i,2j} + \widehat{f}_{2i+2,2j}\right) + \frac{\gamma_2}{2}\left(\widehat{f}_{2i+1,2j-1} + \widehat{f}_{2i+1,2j+1}\right).$$

Аналогічно обчислимо значення $\widehat{f}_{2i,2j+1}$.

Відповідний метод позначимо через $F(C, f, \varphi)$.

Теорема 1 *Для того, щоб метод $F(C, f, \varphi)$ був такий, що*

$$f_{i,j} - F(C, f, \varphi, ih, jh) = f_{i,j} - \widehat{f}_{i,j} = O(h^6), \tag{1.1}$$

необхідно і достатньо, щоб центральні моменти функції φ задовольняли умовам

$$m_{0,4} = -5m_{0,2}, m_{2,2} = -m_{0,2}, \tag{1.2}$$

і при цьому значення коефіцієнтів будуть визначатися наступним чином

$$\alpha_0 = 1 + \frac{5}{8}m_{0,2} - \frac{1}{16}m_{2,2} + \frac{5}{16}m_{0,2}^2 - \frac{1}{32}m_{0,4},$$
$$\alpha_1 = -\frac{2}{3}m_{0,2} + \frac{1}{8}m_{2,2} - \frac{1}{2}m_{0,2}^2 + \frac{1}{24}m_{0,4},$$
$$\alpha_2 = -\frac{1}{16}m_{2,2} + \frac{1}{8}m_{0,2}^2,$$
$$\alpha_3 = -\frac{1}{96}m_{0,4} + \frac{1}{24}m_{0,2} + \frac{1}{16}m_{0,2}^2,$$
$$\beta_0 = -\frac{1}{m_{0,2}}, \beta_1 = 1 + \frac{1}{m_{0,2}},$$
$$\gamma_0 = -\frac{1}{2m_{0,2}}, \gamma_1 = \frac{1}{2} + \frac{1}{2m_{0,2}}, \gamma_2 = \frac{1}{2}.$$

Дійсно, використовуючи формулу Тейлора в околі точки $(2ih, 2jh)$, одержуємо

$$c_{2i,2j} = \int\int_{R^2} f(x+2ih, y+2jh)\varphi(x,y)dxdy =$$

$$= f_{2i,2j}m_{0,0} + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2}\bigg|_{(2i,2j)} m_{2,0} + \frac{h^2}{2}\frac{\partial^2 f}{\partial y^2}\bigg|_{(2i,2j)} m_{0,2} +$$

$$\frac{h^4}{24}\frac{\partial^4 f}{\partial^4 x}\bigg|_{(2i,2j)} m_{4,0} + \frac{h^4}{4}\frac{\partial^4 f}{\partial x^2 \partial y^2}\bigg|_{(2i,2j)} m_{2,2} + \frac{h^4}{24}\frac{\partial^4 f}{\partial y^2}\bigg|_{(2i,2j)} m_{0,4} + O(h^6).$$

Використовуючи формулу реконструкції в точці $(2ih, 2jh)$, отримуємо

$$\hat{f}_{2i,2j} = (\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3)\, f_{2i,2j} +$$

$$+\frac{h^2}{4}\left(2\,\alpha_0 m_{0,2} + \alpha_1\,(4 + 2\,m_{0,2}) + \alpha_2\,(8 + 2\,m_{0,2}) + \right.$$

$$\left. + \alpha_3\,(16 + 2\,m_{0,2})\right)\left(\frac{\partial^2 f}{\partial x^2}\bigg|_{(2i,2j)} + \frac{\partial^2 f}{\partial y^2}\bigg|_{(2i,2j)}\right) +$$

$$+\frac{h^4}{4}\left(2\,\alpha_0 m_{2,2} + \alpha_1\,(4m_{0,2} + \, m_{2,2}) + \alpha_2\,(16 + 8\,m_{0,2} + m_{2,2}) + \right.$$

$$\left. + \alpha_3\,(16\,m_{0,2} + m_{2,2})\right)\,\frac{\partial^4 f}{\partial^2 x \partial^2 y}\bigg|_{(2i,2j)} +$$

$$+\frac{h^4}{24}\left(\,\alpha_0 m_{0,4} + \alpha_1\,(8 + 12\,m_{0,2} + \, m_{0,4}) + \alpha_2\,(16 + 24\,m_{0,2} + \, m_{0,4}) + \right.$$

$$\left. + \alpha_3\,(128 + 48\,m_{0,2} + \, m_{0,4})\right)\left(\frac{\partial^4 f}{\partial x^4}\bigg|_{(2i,2j)} + \frac{\partial^4 f}{\partial y^4}\bigg|_{(2i,2j)}\right) + O(h^6).$$

Таким чином, для того, щоб умова (1.1) виконувалась, треба розв'язати систему рівнянь

$$\begin{cases} \alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = 1, \\ 2\,\alpha_0 m_{0,2} + \alpha_1\,(4 + 2\,m_{0,2}) + \alpha_2\,(8 + 2\,m_{0,2}) + \alpha_3\,(16 + 2\,m_{0,2}) = 0, \\ 2\,\alpha_0 m_{2,2} + \alpha_1\,(4m_{0,2} + \, m_{2,2}) + \alpha_2\,(16 + 8\,m_{0,2} + m_{2,2}) + \alpha_3\,(16\,m_{0,2} + m_{2,2}) = 0, \\ \alpha_0 m_{0,4} + \alpha_1\,(8 + 12\,m_{0,2} + \, m_{0,4}) + \alpha_2\,(16 + 24\,m_{0,2} + \, m_{0,4}) + \\ +\alpha_3\,(128 + 48\,m_{0,2} + \, m_{0,4}) = 0. \end{cases}$$

Розв'язуючи дану систему, отримаємо коефіцієнти $\alpha_0, \alpha_1, \alpha_2, \alpha_3$, наведені в теоремі.

Аналогічним чином, якщо прирівняємо до нуля множники перших похідних різниці $f_{2i+1,2j+1} - \hat{f}_{2i+1,2j+1}$, то отримаємо систему рівнянь, розв'язком якої будуть значення

$$\beta_0 = -\frac{1}{m_{0,2}}, \beta_1 = \frac{1 + m_{0,2}}{m_{0,2}}.$$

При цьому похибка відновлення в точці $((2i+1)h,(2j+1)h)$ буде дорівнювати

$$f_{2i+1,2j+1}-\hat{f}_{2i+1,2j+1}=\frac{h^4}{24}\left(5+\frac{m_{0,4}}{m_{0,2}}\right)\left(\left.\frac{\partial^4 f}{\partial x^4}\right|_{(2i+1,2j+1)}+\left.\frac{\partial^4 f}{\partial y^4}\right|_{(2i+1,2j+1)}\right)+ \tag{1.3}$$

$$\frac{h^4}{4}\left(1+\frac{m_{2,2}}{m_{0,2}}\right)\left.\frac{\partial^4 f}{\partial x^2\partial y^2}\right|_{(2i,2j)}+O(h^6).$$

Нарешті, мінімізуючи похибку реконструкції функції в точці $((2i+1)h,2jh)$, отримаємо

$$\gamma_0=-\frac{1}{2\,m_{0,2}},\gamma_1=\frac{1+m_{0,2}}{2\,m_{0,2}},\gamma_2=\frac{1}{2},$$

і в такому разі маємо

$$f_{2i+1,2j}-\hat{f}_{2i+1,2j}=\frac{h^4}{24}\left(5+\frac{m_{0,4}}{m_{0,2}}\right)\left(\left.\frac{\partial^4 f}{\partial x^4}\right|_{(2i+1,2j)}+\left.\frac{\partial^4 f}{\partial y^4}\right|_{(2i+1,2j)}\right)+ \tag{1.4}$$

$$\frac{h^4}{4}\left(1+\frac{m_{2,2}}{m_{0,2}}\right)\left.\frac{\partial^4 f}{\partial x^2\partial y^2}\right|_{(2i+1,2j)}+O(h^6).$$

Таким чином, вибір моментів згідно (1.2), забезпечує виконання умови (1.1).

У дискретному випадку, коли значення $\varphi(x,y)$ на кожному елементарному квадраті решітки є константою, складність φ визначимо наступним чином

$$\begin{array}{ccccc}0&0&0&0&0\\0&a_{-1,1}&a_{0,1}&a_{1,1}&0\\0&a_{-1,0}&a_{0,0}&a_{1,0}&0\\0&a_{-1,-1}&a_{0,-1}&a_{1,-1}&0\\0&0&0&0&0\end{array}$$

Така функція, що задовольняє умовам теореми, існує і єдина, при цьому ненульові значення декомпозиційного фільтру визначаються матрицею

$$\left[\begin{array}{ccc}-\frac{31}{9360}&-\frac{19}{720}&-\frac{31}{9360}\\-\frac{19}{720}&\frac{1309}{1170}&-\frac{19}{720}\\-\frac{31}{9360}&-\frac{19}{720}&-\frac{31}{9360}\end{array}\right].$$

В цьому випадку центральні моменти функції φ будуть такі

$$m_{0,0}=1,m_{0,2}=\frac{9}{520},m_{2,2}=-\frac{9}{520},m_{0,4}=-\frac{9}{104}$$

і

$$\alpha_0=\frac{877997}{865280},\alpha_1=-\frac{9441}{540800},\alpha_2=\frac{2421}{2163200},\alpha_3=\frac{7101}{4326400},$$

$$\beta_0=-\frac{520}{9},\beta_1=\frac{520}{9},$$

$$\gamma=-\frac{260}{9},\gamma_1=\frac{529}{18},\gamma_2=\frac{1}{2}.$$

Виникає питання, чи можна при тій же асимптотичній точності спростити формули відновлення за рахунок скорочення числа доданків в них? Це можливо, але при цьому носій декомпозиційного фільтру φ необхідно збільшити. Позначимо через $G(C, f, \varphi)$ метод відновлення, який визначається правилом реконструкції

$$\widehat{f}_{2i,2j} = \alpha_0 c_{2i,2j} + \frac{\alpha_1}{4}\left(c_{2i-2,2j} + c_{2i+2,2j} + c_{2i,2j-2} + c_{2i,2j+2}\right) +$$

$$+\frac{\alpha_2}{4}\left(c_{2i-2,2j-2} + c_{2i+2,2j-2} + c_{2i-2,2j-2} + c_{2i+2,2j+2}\right),$$

а відновлення решти значень проводиться по тим же формулам, що й раніше.

Теорема 2 *Для того, щоб метод $G(C, f, \varphi)$ був такий, що*

$$f_{i,j} - G(C, f, \varphi, ih, jh) = f_{i,j} - \widehat{f}_{i,j} = O(h^6),$$

необхідно і достатньо, щоб центральні моменти функції φ були рівні

$$m_{0,0} = 1, m_{0,2} = -\frac{3}{2}, m_{2,2} = \frac{3}{2}, m_{0,4} = \frac{15}{2},$$

і при цьому коефіцієнти формул реконструкції визначаються наступним чином

$$\alpha_0 = \frac{7}{16}, \alpha_1 = \frac{3}{32}, \alpha_2 = \frac{3}{64},$$

$$\beta_0 = -\frac{1}{6}, \beta_1 = \frac{1}{12}, \gamma_0 = -\frac{1}{6}, \gamma_1 = \frac{1}{12}, \gamma_2 = \frac{1}{4}.$$

Без доведення цієї теоремі зазначемо, що у дискретному випадку функція φ існує і єдина. Значення цієї функції визначаються матрицею.

$$\begin{bmatrix} 0 & 0 & \frac{263}{640} & 0 & 0 \\ 0 & \frac{253}{576} & -\frac{1193}{360} & \frac{253}{576} & 0 \\ \frac{263}{640} & -\frac{1193}{360} & \frac{15631}{1440} & -\frac{1193}{360} & \frac{263}{640} \\ 0 & \frac{253}{576} & -\frac{1193}{360} & \frac{253}{576} & 0 \\ 0 & 0 & \frac{263}{640} & 0 & 0 \end{bmatrix}$$

а графік цієї функції має наступний вигляд

Серед двох наведених методів, для практичного використання прийнятнішим є перший, через той факт, що другий декомпозиційний фільтр φ є надмірно контрастуючий.

1.3 Просторові фільтри.

Клас графічних алгоритмів, при реалізації яких вихідний піксел обчислюється по суміжним пікселам, називається просторовими фільтрами (spatial filters). Найпоширенішим просторовим фільтром є згладжуючий фільтр. Процедура згладжування замінює кожен піксел на деяке усереднене його оточення, що дозволяє відфільтрувати випадковий ("білий") шум. Крім того, використання методів обробки даних, заснованих на гладких функціях, більш ефективно, якщо на попередньому етапі згладжувати інформацію, яка надходить. У такому випадку на етапі відновлення доцільно застосовувати оператор зворотного до застосованого оператора згладжування, тобто провести контрастування відновлених даних.

Для перетворення зображення у відтінки сірого з використанням об'ємних ефектів використовую рельєфні фільтри. Рельєфні фільтри дозволяють візуалізувати зміну зображення, тобто показати ту або іншу похідну.

Наприклад, для зображення

використання фільтру

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + 128$$

приводить до наступного ефекту

а фільтр

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} + 128$$

дає таке зображення

Для виявлення динаміки змін елементів зображення в горизонтальному чи вертикальному напряму часто використовують фільтри Собела, які, по суті візуалізують перші похідні $\partial/\partial x$ та $\partial/\partial y$

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} + 128, \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} + 128.$$

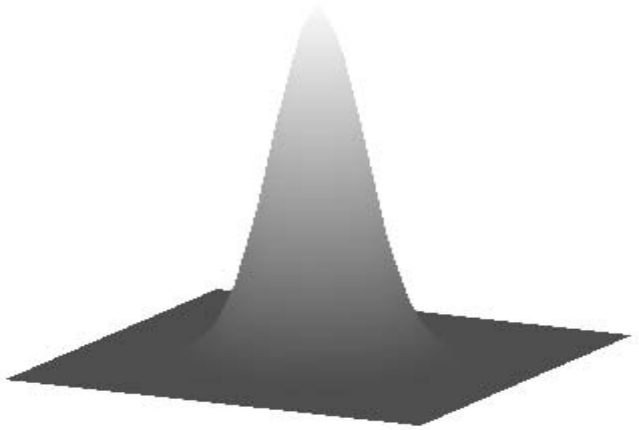
1.3.1 Побудова згладжуючих фільтрів.

В разі наявності на зображенні "білого шуму", для покращення візуального сприйняття, використовують згладжування даного зображення. Існує безліч різноманітних методів згладжування даних. Найпростішим згладжуючим фільтром є усереднення по найближчому оточенню, наприклад, фільтр

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

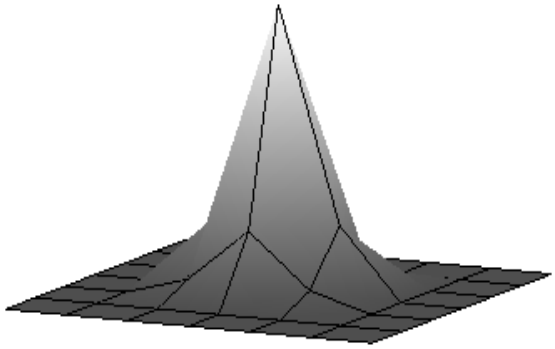
Часто для побудови згладжуючих фільтрів використовують функцію Гауса, наприклад функція

$$z = 2^{-2(x^2+y^2)}$$



породжує згладжуючий фільтр

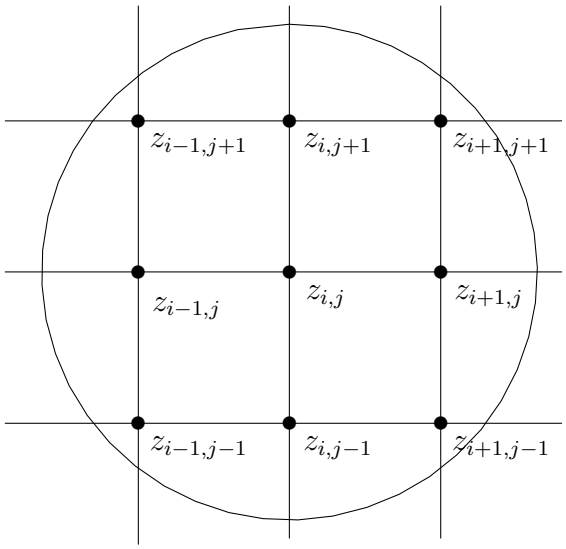
$$G = \frac{1}{36}\begin{bmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{1.5}$$



Дамо приклад використання згладжуючого фільтру

Ми розглянемо одну просту конструкцію згладжування. Вважатимемо, що дані про поверхню задані значеннями $z_{i,j}$ у вузлах решітки (ih, jh) однозв'язної області D.

Для побудови алгоритмів згладжування, разом з кожним значенням $z_{i,j}$, що лежить в середині області D, потрібні сусідні значення $z_{i-1,j-1}$, $z_{i-1,j}$, $z_{i,j-1}$, $z_{i+1,j-1}$, $z_{i-1,j+1}$, $z_{i+1,j+1}$, $z_{i+1,j}$, $z_{i,j+1}$, тобто значення поверхні, відповідні r-околу $(\sqrt{2}h \le r \le 2\,h)$ точки (ih, jh).



У випадку, якщо якесь з цих значення не задане, слід провести поповнення даних. Хай

$$P(x, y) = ax^2 + by^2 + cxy + dx + ey + f$$

квадратична функція двох змінних. Виберемо коефіцієнти a, b, c, d, e і f квадратичної функції з умови мінімізації суми квадратів погрішності, тобто з умови мінімуму величини

$$\sum_{\nu=i-1}^{i+1}\sum_{\mu=j-1}^{j+1}(z_{\nu,\mu}-P(\nu h,\mu h))^2. \tag{1.6}$$

Як згладжене значення братимемо значення екстремальної квадратичної функції в точці (ih, jh).

Для квадратичної функції (1.6) необхідні умови мінімуму співпадають з достатніми. Для визначення коефіцієнтів екстремальної функції необхідно і достатньо узяти часткові похідні від (1.6) і прирівняти їх нулю. При цьому одержимо систему шести лінійних рівнянь з шістьма невідомими. Вирішуючи її, одержуємо квадратичну функцію

$$P^*(x,y)=a^*(x-ih)^2+b^*(-jh)^2+c^*(x-ih)(-jh)+d^*(x-ih)+$$
$$+e^*(-jh)+f^*,$$

де

$$a^*=\frac{1}{6h^2}(z_{i-1,j-1}+z_{i-1,j}+z_{i-1,j+1}+z_{i+1,j-1}+z_{i+1,j}+z_{i+1,j+1}-$$
$$2(z_{i,j-1}+z_{i,j}+z_{i,j+1})),$$

$$b^*=\frac{1}{6h^2}(z_{i-1,j-1}+z_{i,j+1}+z_{i-1,j+1}+z_{i+1,j-1}+z_{i,j-1}+z_{i+1,j+1}-$$
$$2(z_{i-1,j}+z_{i,j}+z_{i+1,j})),$$

$$c^*=\frac{1}{6h^2}\left(z_{i+1,j+1}+z_{i-1,j-1}-z_{i-1,j+1}-z_{i+1,j-1}\right),$$

$$d^*=\frac{1}{6h}\left(z_{i+1,j+1}+z_{i+1,j}+z_{i+1,j-1}-z_{i-1,j+1}-z_{i-1,j}-z_{i-1,j-1})\right),$$

$$e^*=\frac{1}{6h}\left(z_{i+1,j+1}+z_{i-1,j+1}+z_{i,j+1}-z_{i-1,j-1}-z_{i,j-1}-z_{i-1,j-1})\right),$$

$$f^*=z_{i,j}+\frac{2}{9}(z_{i-1,j}+z_{i,j+1}+z_{i+1,j}+z_{i,j-1}-4zi,j)-$$
$$\frac{1}{9}(z_{i-1,j+1}+z_{i+1,j+1}+z_{i+1,j-1}+z_{i-1,j-1}-4z_{i,j})=$$
$$z_{i,j}+\frac{1}{9}\left(2\Delta^2 z_{i,j}-\widetilde{\Delta}^2 z_{i,j}\right).$$

Таким чином згладженим значенням можна рахувати значення $\widetilde{z}_{i,j}$, що визначається рівністю

$$\widetilde{z}_{i,j}=z_{i,j}+\frac{1}{9}\left(2\Delta^2 z_{i,j}-\widetilde{\Delta}^2 z_{i,j}\right),$$

де

$$\Delta^2 z_{i,j} = z_{i-1,j} + z_{i,j+1} + z_{i+1,j} + z_{i,j-1} - 4zi, j$$

і

$$\widetilde{\Delta}^2 z_{i,j} = z_{i-1,j+1} + z_{i+1,j+1} + z_{i+1,j-1} + z_{i-1,j-1} - 4z_{i,j}.$$

В цьому випадку оператор згладжування породжений фільтром

$$S = (s_{i,j})_{i,j=-1}^{1} = \frac{1}{9}\begin{bmatrix} -1 & 2 & -1 \\ 2 & 5 & 2 \\ -1 & 2 & -1 \end{bmatrix}. \tag{1.7}$$

1.3.2 Побудова методів контрастування.

Ясна річ, використання згладжування тягне за собою і зворотню операцію - контрастування. Для побудови методів контрастування нам знадобиться одне твердження.

Через ℓ_2^2 позначимо лінійний простір всіх обмежених двовимірних послідовностей (масивів)

$$A = \{\widetilde{a}_{i,j}\}_{(i,j)\in Z^2}$$

з нормою

$$\|A\| = \left(\sum_{i,j\in Z} a_{i,j}^2\right)^{1/2}$$

і покладемо

$$|A| = \left|\sum_{i,j\in Z} a_{i,j}\right|.$$

Хай L лінійний оператор, який відображає простір X у простір Y. Як завжди, нормою оператора L називатимемо величину

$$\|L\|_{X\to} = \sup_{\|F\|_X \le 1} \|L(F)\|_Y.$$

Позначимо через $C = A \otimes B$ згортку масивів (матриць) A і B, тобто масив C такий, що

$$c_{i,j} = \sum_{\nu,\mu} a_{\nu,\mu} b_{\nu-i,\mu-j}.$$

Розглянемо рівняння

$$E \otimes X = F, \tag{1.8}$$

де $|E| \neq 0$, тоді

$$\frac{E}{|E|} \otimes X = \frac{F}{|E|}$$

або, що те ж

$$\widetilde{E} \otimes X = \Phi, \tag{1.9}$$

де

$$\widetilde{E} = \frac{E}{|E|}$$

і

$$\Phi = \frac{F}{|E|}.$$

Основою для побудови контрастуючих фільтрів є наступне твердження.

Теорема 3 *Якщо для будь-якого натурального n виконується нерівність*

$$\|I - \widetilde{E}\|_{\ell_2^2 \to \ell_2^2} < 1, \tag{1.10}$$

то має місце співвідношення

$$X_n = C_n^1\Phi - C_n^2\widetilde{E} \otimes \Phi + C_n^3\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^n C_n^n \widetilde{E}^n \otimes \Phi + \varepsilon_n, \tag{1.11}$$

де ε_n таке, що

$$\widetilde{E} \otimes \varepsilon_n = (I - \widetilde{E})^{n+1} \otimes \Phi. \tag{1.12}$$

Дійсно, якщо

$$X_n = C_n^1\Phi - C_n^2\widetilde{E} \otimes \Phi + C_n^3\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^n C_n^n \widetilde{E}^n \otimes \Phi + \varepsilon_n$$

то

$$\widetilde{E} \otimes X_n = \widetilde{E} \otimes (C_n^1\Phi - C_n^2\widetilde{E} \otimes \Phi + C_n^3\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^n C_n^n \widetilde{E}^n \otimes \Phi + \varepsilon_n) =$$

$$= C_n^1\widetilde{E} \otimes \Phi - C_n^2\widetilde{E}^2 \otimes \Phi + C_n^4\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^n C_n^n \widetilde{E}^{n+1} \otimes \Phi + \widetilde{E} \otimes \varepsilon_n.$$

Звідси і з (1.12) відразу одержуємо

$$\widetilde{E} \otimes X_n = C_n^1\widetilde{E} \otimes \Phi - C_n^2\widetilde{E}^2 \otimes \Phi + C_n^4\widetilde{E}^2 \otimes \Phi + \ldots$$

$$+(-1)^n C_n^n \widetilde{E}^{n+1} \otimes \Phi + (I - \widetilde{E})^{n+1} \otimes \Phi =$$

$$= C_n^1\widetilde{E} \otimes \Phi - C_n^2\widetilde{E}^2 \otimes \Phi + C_n^4\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^n C_n^n \widetilde{E}^{n+1} \otimes \Phi +$$

$$+\Phi - C_n^1\widetilde{E} \otimes \Phi + C_n^2\widetilde{E}^2 \otimes \Phi - C_n^4\widetilde{E}^2 \otimes \Phi + \ldots + (-1)^{n+1} C_n^n \widetilde{E}^{n+1} \otimes \Phi = \Phi.$$

Помітимо, що, якщо виконується умова (1.10), то

$$\|\varepsilon_n\| \le |\widetilde{E}^{-1}|\ \|I - \widetilde{E}\|^{n+1}|\Phi|,$$

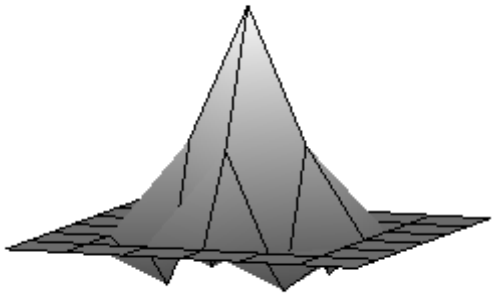
і в цьому випадку рішення можна записати в операторному вигляді

$$X = \lim_{n\to\infty}(I - (I - \widetilde{E})^n)\Phi.$$

Використовуємо цей результат для побудови оператора контрастування, що повертає зображення, змінене в результаті застосування оператора згладжування.

Як приклад розглянемо побудову контрастуючого фільтру псевдозворотнього до згладжуючого фільтру (1.7), побудованому раніше

$$S = \begin{bmatrix} -\frac{1}{9} & \frac{2}{9} & -\frac{1}{9} \\ \frac{2}{9} & \frac{5}{9} & \frac{2}{9} \\ -\frac{1}{9} & \frac{2}{9} & -\frac{1}{9} \end{bmatrix}$$



Відповідний контрастуючий фільтр S^{-1} повинен задовольняти умові

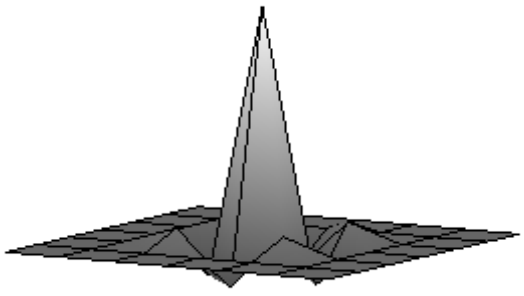
$$S \otimes S^{-1} = I,$$

де

$$I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Застосовуючи формулу (1.11), отримуємо наближене значення S^{-1}. Перша ітерація дає контрастуючий фільтр

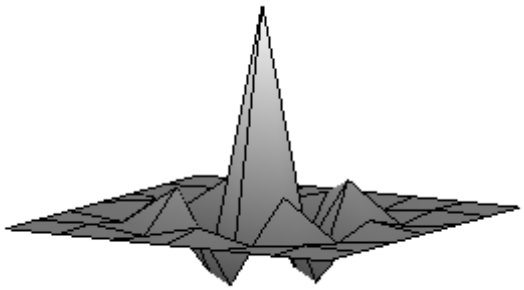
$$S^{-1,1} = 2I - S \otimes I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{9} & -\frac{2}{9} & \frac{1}{9} & 0 \\ 0 & -\frac{2}{9} & \frac{13}{9} & -\frac{2}{9} & 0 \\ 0 & \frac{1}{9} & -\frac{2}{9} & \frac{1}{9} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Застосування другої ітерації дозволяє побудувати контрастуючий фільтр, що по-

вертає згладжені дані з вищою точністю

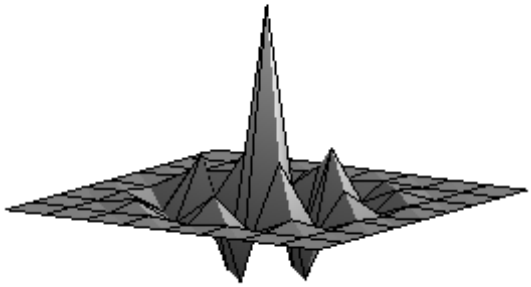
$$S^{-1,2} = 3I - 3S \otimes I + S^2 \otimes I = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{81} & -\frac{4}{81} & \frac{2}{27} & -\frac{4}{81} & \frac{1}{81} & 0 \\ 0 & -\frac{4}{81} & \frac{25}{81} & -\frac{14}{27} & \frac{25}{81} & -\frac{4}{81} & 0 \\ 0 & \frac{2}{27} & -\frac{14}{27} & \frac{17}{9} & -\frac{14}{27} & \frac{2}{27} & 0 \\ 0 & -\frac{4}{81} & \frac{25}{81} & -\frac{14}{27} & \frac{25}{81} & -\frac{4}{81} & 0 \\ 0 & \frac{1}{81} & -\frac{4}{81} & \frac{2}{27} & -\frac{4}{81} & \frac{1}{81} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Результат третьої ітерації дає фільтр

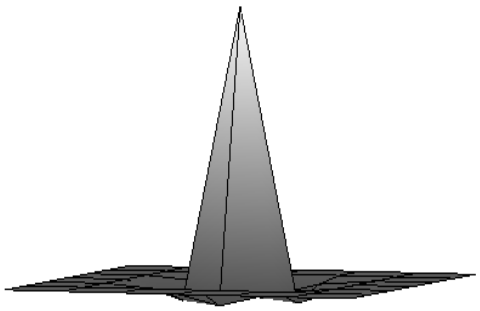
$$S^{-1,3} = 4I - 6S \otimes I + 4S^2 \otimes I - S^3 \otimes I$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{729} & -\frac{2}{243} & \frac{5}{243} & -\frac{20}{729} & \frac{5}{243} & -\frac{2}{243} & \frac{1}{729} & 0 \\ 0 & -\frac{2}{243} & \frac{5}{81} & -\frac{14}{81} & \frac{58}{243} & -\frac{14}{81} & \frac{5}{81} & -\frac{2}{243} & 0 \\ 0 & \frac{5}{243} & -\frac{14}{81} & \frac{50}{81} & -\frac{226}{243} & \frac{50}{81} & -\frac{14}{81} & \frac{5}{243} & 0 \\ 0 & -\frac{20}{729} & \frac{58}{243} & -\frac{226}{243} & \frac{1777}{729} & -\frac{226}{243} & \frac{58}{243} & -\frac{20}{729} & 0 \\ 0 & \frac{5}{243} & -\frac{14}{81} & \frac{50}{81} & -\frac{226}{243} & \frac{50}{81} & -\frac{14}{81} & \frac{5}{243} & 0 \\ 0 & -\frac{2}{243} & \frac{5}{81} & -\frac{14}{81} & \frac{58}{243} & -\frac{14}{81} & \frac{5}{81} & -\frac{2}{243} & 0 \\ 0 & \frac{1}{729} & -\frac{2}{243} & \frac{5}{243} & -\frac{20}{729} & \frac{5}{243} & -\frac{2}{243} & \frac{1}{729} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

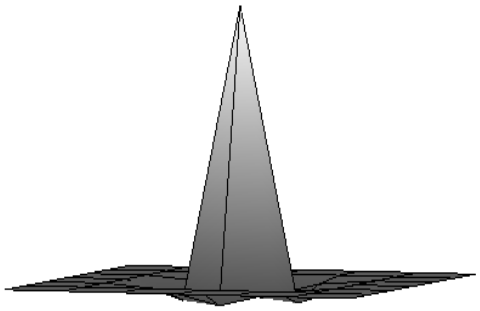


Приведемо контрастуючі фільтри для згладжуючого фільтра Гауса (1.5).

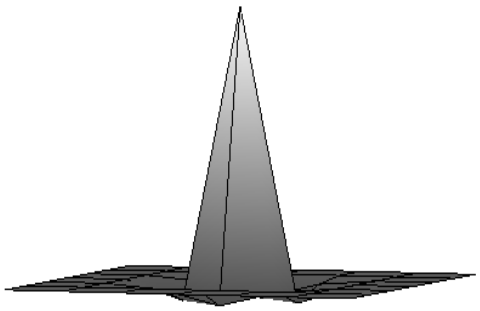
$$G^{-1,1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{36} & -\frac{1}{9} & -\frac{1}{36} & 0 \\ 0 & -\frac{1}{9} & \frac{14}{9} & -\frac{1}{9} & 0 \\ 0 & -\frac{1}{36} & -\frac{1}{9} & -\frac{1}{36} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$G^{-1,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{1296} & \frac{1}{162} & \frac{1}{72} & \frac{1}{162} & \frac{1}{1296} & 0 \\ 0 & \frac{1}{162} & -\frac{11}{324} & -\frac{2}{9} & -\frac{11}{324} & \frac{1}{162} & 0 \\ 0 & \frac{1}{72} & -\frac{2}{9} & \frac{23}{12} & -\frac{2}{9} & \frac{1}{72} & 0 \\ 0 & \frac{1}{162} & -\frac{11}{324} & -\frac{2}{9} & -\frac{11}{324} & \frac{1}{162} & 0 \\ 0 & \frac{1}{1296} & \frac{1}{162} & \frac{1}{72} & \frac{1}{162} & \frac{1}{1296} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$G^{-1,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{46656} & -\frac{1}{3888} & -\frac{17}{15552} & -\frac{11}{5832} & -\frac{17}{15552} & -\frac{1}{3888} & -\frac{1}{46656} & 0 \\ 0 & -\frac{1}{3888} & 0 & \frac{5}{432} & \frac{8}{243} & \frac{5}{432} & 0 & -\frac{1}{3888} & 0 \\ 0 & -\frac{17}{15552} & \frac{5}{432} & -\frac{43}{1728} & -\frac{619}{1944} & -\frac{43}{1728} & \frac{5}{432} & -\frac{17}{15552} & 0 \\ 0 & -\frac{11}{5832} & \frac{8}{243} & -\frac{619}{1944} & \frac{1580}{729} & -\frac{619}{1944} & \frac{8}{243} & -\frac{11}{5832} & 0 \\ 0 & -\frac{17}{15552} & \frac{5}{432} & -\frac{43}{1728} & -\frac{619}{1944} & -\frac{43}{1728} & \frac{5}{432} & -\frac{17}{15552} & 0 \\ 0 & -\frac{1}{3888} & 0 & \frac{5}{432} & \frac{8}{243} & \frac{5}{432} & 0 & -\frac{1}{3888} & 0 \\ 0 & -\frac{1}{46656} & -\frac{1}{3888} & -\frac{17}{15552} & -\frac{11}{5832} & -\frac{17}{15552} & -\frac{1}{3888} & -\frac{1}{46656} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



Приведемо приклад використання контрастуючого фільтру

На завершення параграфа, присвяченого створенню просторових фільтрів, розглянемо реалізацію ефекту акварелізації. Акварельний фільтр перетворить зображення, і після обробки воно виглядає так, як ніби написано аквареллю. Для отримання такого ефекту використовується метод медіанного усереднення кольору в кожній точці. Значення кольору кожного піксела і його 24 сусідів поміщаються в список і сортуються від меншого до більшого. Медіанне (тринадцяте) значення кольору в списку привласнюється центральному пікселу. Після цього для виділення межі переходів кольорів, до одержаного зображення застосовується контрастуючий фільтр. Результуюче зображення нагадує акварельний живопис.

Це лише один приклад, який показує, як можна об'єднувати різні методи обробки зображень і добиватися незвичайних візуальних ефектів.

1.4 Побудова частотного аналізу зображення.

Традиційно для частотного аналізу сигналу, і зображень в тому числі, використовується перетворення Фур'є (що реалізовано в форматі JPEG), але останнім часом для цієї мети використовується апарат сплесків (вейвлетів, wavelets).

У даному параграфі ми проілюструємо методику частотного аналізу, який використовується, наприклад, в JPEG2000, на прикладі найпростішого wavelet-фільтра – фільтра Хаара (Haar).

Через $\mathbf{H}$ позначимо четвірку фільтрів (фільтр Хаара)

$$H = \frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad H^{+-} = \frac{1}{4}\begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix},$$

$$H^{\pm} = \frac{1}{4}\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}, \quad H^{\times} = \frac{1}{4}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

а через $\mathbf{Q} = \mathbf{H} \otimes \mathbf{P}$ позначимо згортку $\mathbf{H}$ з масивом $\mathbf{P}$, тобто

$$q_{i,j} = \frac{1}{4}\left(p_{2i,2j} + p_{2i+1,2j} + p_{2i,2j+1} + p_{2i+1,2j+1}\right);$$

$$q_{i,j}^{+-} = \frac{1}{4}\left(p_{2i,2j} - p_{2i+1,2j} + p_{2i,2j+1} - p_{2i+1,2j+1}\right);$$

$$q_{i,j}^{\pm} = \frac{1}{4}\left(p_{2i,2j} + p_{2i+1,2j} - p_{2i,2j+1} - p_{2i+1,2j+1}\right);$$

$$q_{i,j}^{\times} = \frac{1}{4}\left(p_{2i,2j} - p_{2i+1,2j} - p_{2i,2j+1} + p_{2i+1,2j+1}\right).$$

Фільтр **H** утворює повну систему, тобто знаючи коефіцієнти $q_{i,j}$, $q_{i,j}^{+-}$, $q_{i,j}^{\pm}$ і $q_{i,j}^{\times}$, можна відновити початкові значення $p_{i,j}$. Дійсно, легко бачити, що

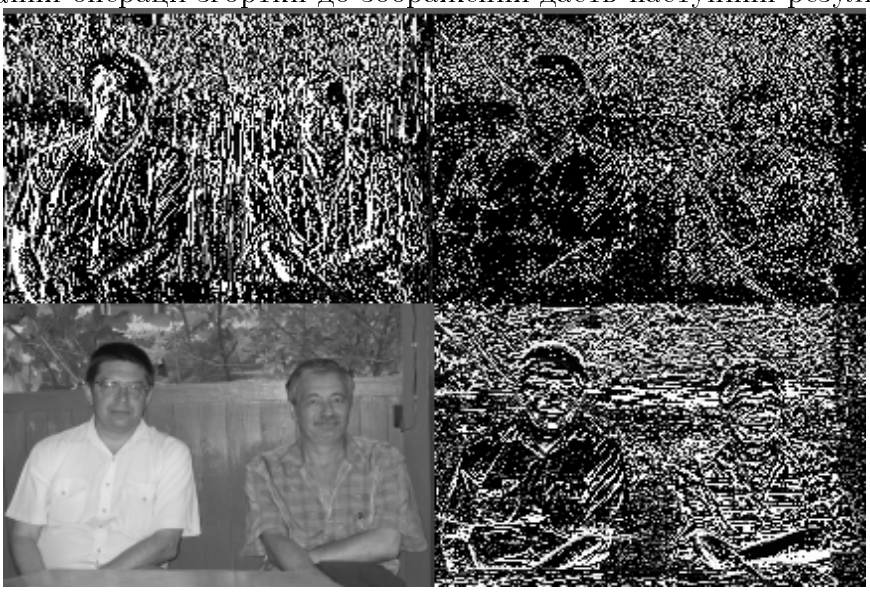
$$p_{2i,2j} = q_{i,j} + q_{i,j}^{+-} + q_{i,j}^{\pm} + q_{i,j}^{\times};$$

$$p_{2i,2j+1} = q_{i,j} + q_{i,j}^{+-} - q_{i,j}^{\pm} - q_{i,j}^{\times};$$

$$p_{2i+1,2j} = q_{i,j} - q_{i,j}^{+-} + q_{i,j}^{\pm} - q_{i,j}^{\times};$$

$$p_{2i+1,2j+1} = q_{i,j} - q_{i,j}^{+-} - q_{i,j}^{\pm} + q_{i,j}^{\times}.$$

Застосування операції згортки до зображення дасть наступний результат



Ясно, що елементи згортки ядра H із зображенням є початковим зображенням зменшеним в чотири рази, елементи згортки з ядром H^{+-} дають відмінність точок початкового зображення по горизонталі, із згортки з ядром $H^{\pm}$ – по вертикалі. Коефіцієнти $q_{i,j}^{\times}$ акумулюють відмінності, що не увійшли до інших випадків.

Послідовне застосування процедури згортки фільтру з множиною середніх значень називається каскадною схемою фільтрації. Послідовне застосування фільтру Хаара до попереднього результату матиме вигляд

При зберіганні одержаних частотних характеристик потрібно мати на увазі, що коефіцієнти одержані на вищому рівні декомпозиції більш важливі, ніж ті, які одержані раніше. Тому для високочастотних коефіцієнтів можна застосовувати достатньо грубе квантування, але з кожним рівнем квантування повинне бути точніше.

1.5 Використання сплесків для частотного аналізу зображень.

Сплески (wavelets) з'явилися на початку 1980-х років на стику теорії функцій, функціонального аналізу, обробки сигналів і зображень, квантової теорії поля. Розвиток теорії сплесків пов'язаний з іменами А. Гроссмана, Же. Морлета, Же. Стремберга, Я. Мейєра, З. Малла, І. Добеші та багатьох ін.

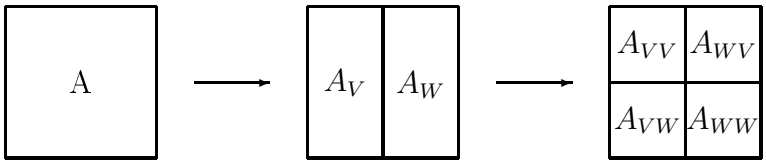
Подібно аналізу Фур'є теорія сплесків знаходить застосування в дослідженні частотних характеристик функції f за допомогою сплеск-перетворення. При описі функцій з скінченним носієм, сигналів з змінними в часі частотами за допомогою аналізу Фур'є виникають труднощі, які пояснюються властивостями тригонометричних функцій на осі R. Наприклад, перетворення Фур'є функції f у випадку δ-функції Дірака, що має як носій єдину точку, вироджується в функцію $\widehat{\delta}(\omega) = 1$, поширену на всю числову вісь. При обробці сигналів із змінними частотами аналіз Фур'є не дозволяє виділити частоти, що становлять сигнал в околі довільного моменту часу. Апарат сплесків дозволяє автоматично здійснювати локалізацію як за часом, так і в частотній області.

Не заглиблюючись в математичну конструкцію сплесків, опишемо один з найпопулярніших методів побудови сплескових фільтрів для обробки двовимірних даних (зображень). Цей метод був запропонований S.Mallat і називається схемою Малла.

Як базис простору $\mathbf{V}^0$ візьмемо сукупність функцій $\varphi(x-n)\varphi(-m)$.

Таким чином в цьому просторі базис, породжений функцією $\varphi(x,y) = \varphi(x)\varphi(y)$ та ортогональним доповненням $\mathbf{W}^0$ можна взяти функції $\psi_1(x,y) = \varphi(x)\psi(y)$, $\psi_2(x,y) = \psi(x)\varphi(y)$, $\psi_3(x,y) = \psi(x)\psi(y)$.

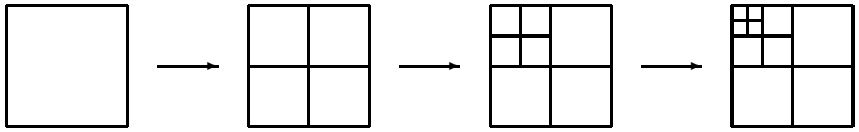
Схематично процедура сплеск(вейвлет)-перетворення двовимірних даних простору $\mathbf{V}^1$ проілюстрована на наступному рисунку.



Тобто, на першому етапі, розподіляємо коефіцієнти розкладання по зрушеннях масштабуючої функції простору $\mathbf{V}^1$ у вигляді матриці (у загальному випадку нескінченної). Застосовуючи послідовно до коефіцієнтів кожного рядка одновимірний алгоритм розкладання на простір V і ортогональне доповнення W, одержуємо дві матриці A_V (низькочастотний домен) і A_W (високочастотний домен), які відповідають цим просторам. На другому етапі цей же алгоритм застосовується до стовпців кожної з одержаних матриць. В результаті, з кожної матриці знов виходить дві. Результат застосування масштабуючого фільтру до матриці A_V дасть матрицю A_{VV}, сплескового фільтру - матрицю A_{VW}. Результатом аналогічного перетворення матриці A_W є матриці A_{WV} і A_{WW}. Простір сплесків в цьому випадку можна записати у вигляді

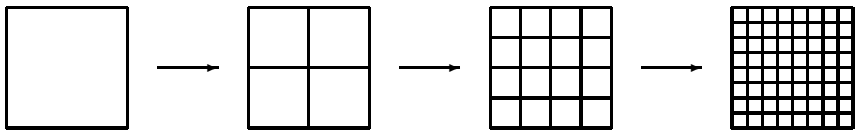
$$\mathbf{W}^0 = \left(W^0 \otimes W^0\right) \oplus \left(W^0 \otimes V^1\right) \oplus \left(V^0 \otimes W^1\right).$$

Застосовуючи послідовно описаний алгоритм до A_{VV}, одержуємо каскадний алгоритм вейвлет перетворення двовимірних даних.

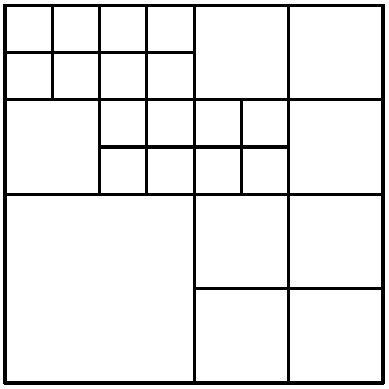


Описана схема не є єдиною. Серед великого набору методів обробки двовимірних даних приведемо ще один - використання вейвлет-пакетів. На відміну від каскадного алгоритму, при використанні вейвлет-пакетів одновимірне вейвлет-перетворення застосовується до кожного частотного домену (тобто до кожної з матриць A_{VV}, A_{VW}, A_{WV} і A_{WW}).

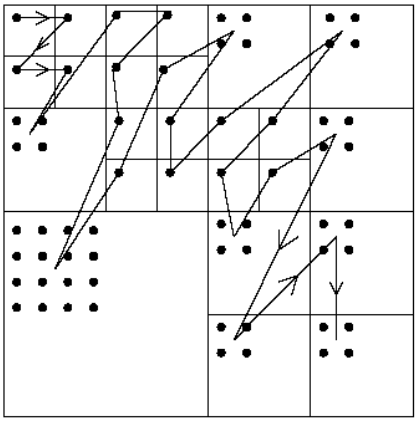
Схема повного пакету може бути проілюстрована таким чином.



Як правило, на практиці до кожного подальшого високочастотного домена застосовується інший базис. Крім того, повний пакет сплесків рідко використовується, зважаючи на великі часові та обчислювальні витрати. Доцільно використовувати вибіркову, а не повну пакетну схему. Помітимо, що, в цьому випадку, схема відновлення істотно ускладнюється, оскільки при відновленні значення з кожного частотного домена потрібно брати різне число коефіцієнтів. Наприклад для схеми



відновлення проходитиме таким чином

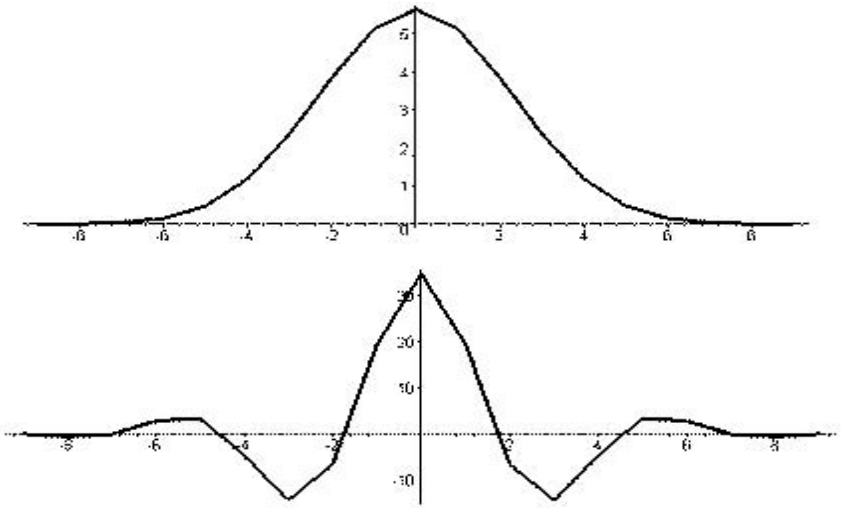


Як приклад одновимірних сплескових фільтрів, наведемо класичну біортогональну пару 3-5 Коена-Добеші- Фово.

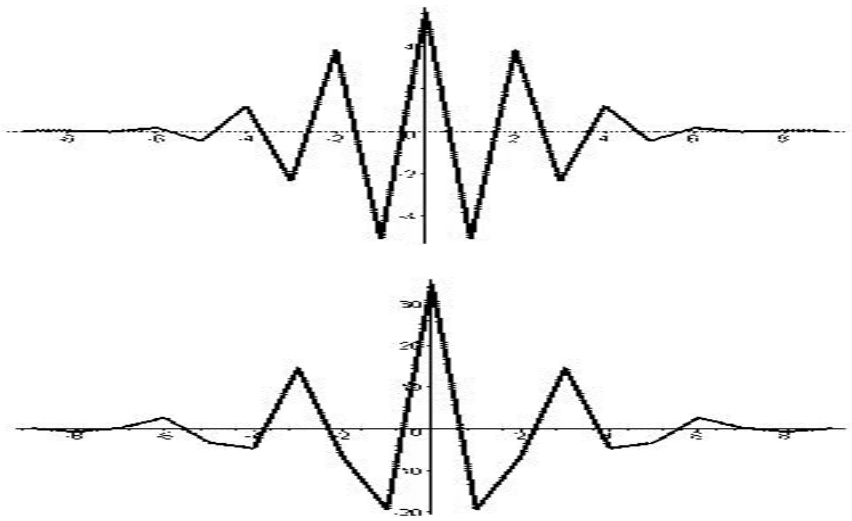
$$a_{-1} = \frac{\sqrt{2}}{4}, a_0 = \frac{\sqrt{2}}{2}, a_1 = \frac{\sqrt{2}}{4},$$

$$b_{-2} = -\frac{\sqrt{2}}{8}, b_{-1} = \frac{\sqrt{2}}{4}, b_0 = \frac{3\sqrt{2}}{4}, b_1 = \frac{\sqrt{2}}{4}, b_2 = -\frac{\sqrt{2}}{8}.$$

На малюнку приведена біортогональна пара масштабуючих функцій



і відповідних сплескових функцій



Прямий хід швидкого сплеск перетворення масиву $F = \{f_i\}$ у точці $2i$ дає згортка масиву з фільтром

$$a_{-1} = \frac{\sqrt{2}}{4}, a_0 = \frac{\sqrt{2}}{2}, a_1 = \frac{\sqrt{2}}{4},$$

у точці $2i+1$ з фільтром

$$b_{-2} = -\frac{\sqrt{2}}{8}, b_{-1} = -\frac{\sqrt{2}}{4}, b_0 = \frac{3\sqrt{2}}{4}, b_1 = -\frac{\sqrt{2}}{4}, b_2 = -\frac{\sqrt{2}}{8},$$

тобто

$$\widehat{f}_{2i} = \frac{\sqrt{2}}{4}\left(f_{2i-1} + 2f_{2i} + f_{2i+1}\right),$$

$$\widehat{f}_{2i+1} = -\frac{\sqrt{2}}{8}\left(f_{2i-1} + 2f_{2i} - 6f_{2i+1} + 2f_{2i+2} + f_{2i+3}\right).$$

Зворотнє сплеск-перетворення в точці $2i$ дає згортка масиву сплеск-перетворення з фільтром

$$u_{-2} = -\frac{\sqrt{2}}{8}, u_{-1} = -\frac{\sqrt{2}}{4}, u_0 = \frac{3\sqrt{2}}{4}, u_1 = -\frac{\sqrt{2}}{4}, u_2 = -\frac{\sqrt{2}}{8},$$

у точці $2i+1$ з фільтром

$$v_{-1} = \frac{\sqrt{2}}{4}, v_0 = \frac{\sqrt{2}}{2}, v_1 = \frac{\sqrt{2}}{4},$$

тобто

$$f_{2i} = -\frac{\sqrt{2}}{8}\left(\widehat{f}_{2i-2} + 2\widehat{f}_{2i-1} - 6\widehat{f}_{2i} + 2\widehat{f}_{2i+1} + \widehat{f}_{2i+2}\right),$$

$$f_{2i+1} = \frac{\sqrt{2}}{4}\left(\widehat{f}_{2i} + 2\widehat{f}_{2i+1} + \widehat{f}_{2i+2}\right).$$

Розділ 2

Методи стиску даних.

Дані, що зберігаються на різних носіях і передаються по каналах зв'язку, як правило, мають значну надмірність. Використання алгоритмів стиску інформації дозволяє підвищити в кілька разів швидкість обміну по каналах зв'язку і в стільки ж разів економити об'єм дискового простору. На сьогоднішній день існує безліч підходів до стиску інформації і алгоритмів, які їх реалізують. Перш за все, це статистичні і словарні методи.

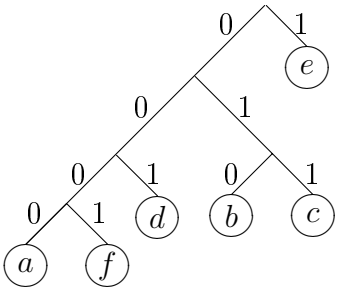
2.1 Метод Хаффмана

Метод стиску Хаффмана (Huffman, 1952) – це загальна схема стиску інформації, для дискретних даних різної природи.

Ідея методу полягає в заміні даних ефективнішими кодами, і складається з привласнення бінарного коду кожній унікальній величині. Коротші коди використовуються для тих величин, що зустрічаються частіше. Ці привласнення зберігаються в таблиці перекодування, яка завантажується в декодуючу програму перед самими кодами.

Наприклад, в серії *abccddeeeeeeecbbf* є шість унікальних величин. Частоти їх запису дорівнюють *a:1 b:3 c:3 d:2 e:7 f:1.*

Щоб сформувати мінімальний код, використаємо бінарне дерево



Основний алгоритм об'єднує разом всі елементи, що з'являються якнайрідше, потім пара розглядається як один елемент і їх частоти об'єднуються. Це повторюється до тих пір, поки всі елементи не об'єднаються в пари.

Для даного прикладу найрідше використовуються значення a і f, які складають першу пару – a привласнюється 0-а гілка, а f – відповідно 1-а. Це значить, що в

кодах цих символів молодшими бітами будуть відповідно 0 і 1. Старші біти виходять по мірі побудови дерева. Наступна частота – 2. Тому одержана пара об'єднується з символом, який має частоту 2, тобто з d. Початковій парі привласнюється 0 гілка нового дерева, а елементу d гілка 1. Таким чином, на новому етапі елемент a має код 00, а елемент f код 01. Продовжуючи побудову дерева, одержуємо, що чим частіше зустрічається елемент, тим коротше його код.

Таблиця кодів для даного випадку виглядатиме так:

e:7	0	1		
b:3	0	1	0	
c:3	0	1	1	
d:2	0	0	1	
a:1	0	0	0	0
f:1	0	0	0	1

Хоча даний приклад і ілюструє метод Хаффмана, код, одержуваний в результаті використання такого алгоритму, не є найкоротшим.

Спорідненим методом для кодування Хаффмана є кодування Шеннона-Фано (Shannon-Fano method), яке здійснюється таким чином:

- Ділимо множину символів на дві підмножини так, щоб сума вірогідності появи символів однієї підмножини була приблизно рівна сумі вірогідності появи символів іншого. Для лівої підмножини кожному символу приписуємо "0", для правого - "1".
- Повторюємо попередній крок до тих пір, поки всі підмножини не будуть складатися з одного елементу.

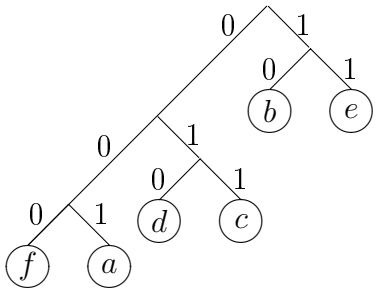
Розглянемо цей підхід на нашому прикладі. Розташуємо елементи алфавіту відповідно до їх частотних характеристик: *e:7 b:3 c:3 d:2 a:1 f:1.* Розіб'ємо одержану таблицю на дві частини так, щоб суми частот в кожній з цих частин були приблизно рівні, і елементам, які стоять в першій таблиці привласнимо значення 1, а в другій – 0.

e:7	1
b:3	1
c:3	0
d:2	0
a:1	0
f:1	0

Далі, з кожною з одержаних таблиць проведемо таке ж перетворення, і так до тих пір, поки не одержимо набір таблиць, що складаються з одного елементу:

e:7	1	1	
b:3	1	0	
c:3	0	1	1
d:2	0	1	0
a:1	0	0	1
f:1	0	0	0

Відповідне бінарне дерево виглядає таким чином:



Код Шеннона-Фано не завжди є оптимальним, кращих результатів дозволяє добитися власне метод Хаффмана. При його використанні кожен етап кодування полягає у відсіканні двох символів, що найбільш часто зустрічаються (що повністю співпадає з приведеним раніше прикладом). Алгоритм створення коду Хаффмана називається знизу-вгору, а Шеннона-Фано - зверху-вниз.

Схема Хаффмана досягає ступеня стиску 1:8 і потребує точної статистики опису частот використовування того або іншого елементу. Без точної статистики кінцевий файл не стає набагато менше за початковий, тому для забезпечення ефективного ступеня стиску ця схема часто реалізується в два проходи. За перший прохід створюється статистична модель, за другий – кодуються дані. У результаті компресія і декомпресія по Хаффману - порівняльно повільні процеси. Крім того, оскільки біти стискаються без урахування границь байта, єдиний спосіб, за допомогою якого дешифратор може дізнатися про завершення кода – досягти кінця гілки, інакше (тобто за наявності збою) дешифратор починає з середини і решта даних стає безглуздою.

Приведемо ще одне зауваження: алгоритм Хаффмана не годиться для адаптованих моделей. На це є наступні причини:

По-перше, всякий раз при зміні моделі необхідно змінювати і весь набір кодів. Хоча ефективні алгоритми роблять це за рахунок невеликих додаткових витрат, ним все одно потрібне місце для розміщення дерева кодів. Якщо його використовувати в адаптованому кодуванні, то для різної вірогідності розподілу і відповідної безлічі кодів будуть потрібні свої класи умов для прогнозу символу. Оскільки моделі можуть бути досить громіздкі, то зберігання всіх дерев кодів стає надмірно дорогим. Добре наближення до кодування Хаффмана може бути досягнуте застосуванням різновиду дерев, що розширяються. При цьому, представлення дерева достатньо компактне, щоб зробити можливим його застосування в моделях, що мають декілька сотень класів умов.

По-друге, метод Хаффмана неприйнятний в адаптованому кодуванні, оскільки виражає значення $-\log p$ цілим числом бітів. Це особливо недоречно, коли один символ має високу вірогідність (що і є частим випадком в складних адаптованих моделей). Якнайменший код, який може бути отриманий методом Хаффмана має 1 біт у довжину, хоча часто бажано використовувати менший. Наприклад, "o"в контексті "to be or not to be"можна закодувати в 0.014 біти. Код Хаффмана перевищує необхідний вихід в 71 разів, роблячи точний прогноз даремним.

Цю проблему можна подолати блокуванням символів, які зустрічаються досить рідко. Проте, це вносить свої проблеми, пов'язані з розширенням алфавіту (який

тепер є множиною всіх можливих блоків).

2.2 Словарні методи стиску LZ.

Найпоширеніші на сьогодняшній день методи стиску засновані на словарних алгоритмах.

Існують дві гілки словарних алгоритмів. Перша бере початок від алгоритму LZ77 і ґрунтується на заміні рядка символів в потоці на посилання на такий же рядок, що раніше зустрівся, з довжиною замінюваного рядка. Друга походить від алгоритму LZ78. У алгоритмах цієї гілки по ходу роботи заповнюється словник. Якщо у вхідному потоці зустрівся рядок, вже присутній в словнику, він замінюється на відповідний номер елемента словника. В даний час LZ77-подібні алгоритми практично не використовуються самостійно - створена безліч вдалих комбінованих алгоритмів, де вихідний потік LZ77-подібного алгоритму стискається контекстно-вільним алгоритмом вірогідності. Як правило, це (у порядку спадання швидкодії і підвищення ступеня стиску) алгоритми Шеннона-Фоно, Хаффмана, арифметичний стиск. Саме ці комбіновані алгоритми успішно використовуються в більшості сучасних універсальних архіваторів (ARJ, PKZIP, LHA, RAR). Сімейство LZ78 складається з алгоритму LZW і його незначних модифікацій. LZC - чисто практична реалізація LZW-алгоритму, вживана в програмі COMPRESS, використовуваній в системі UNIX і в графічному форматі GIF.

Своєю появою він зобов'язаний двом дослідникам з Ізраїлю: Якобу Зіву і Абраму Лемпелу (Jacob Ziv & Abraham Lempel). У 1977 році вони опублікували роботу [18]. Її поява ознаменувала початок нової ери в стиску інформації. Практично всі сучасні комп'ютерні програми-архіватори використовують ту або іншу модифікацію алгоритму LZ.

Основна ідея алгоритму LZ полягає у тому, що друге і подальші входження деякого рядка символів в повідомлення замінюються посиланням на її першу появу в повідомленні.

LZ77 використовує вже проглянуту частину повідомлення як словник. Щоб добитися стиску, він намагається замінити черговий фрагмент повідомлення на покажчик у вміст словника. Як модель даних LZ77 використовує "ковзаюче"по повідомленню вікно, розділене на дві нерівні частини. Перша, велика за розміром, включає вже проглянуту частину повідомлення. Друга, набагато менша, є буфером, що містить ще не закодовані символи вхідного потоку. Звичайно розмір вікна складає декілька кілобайт. Буфер набагато менше, зазвичай не більш сто байтів. Алгоритм намагається знайти у словнику фрагмент, співпадаючий з вмістом буфера.

Алгоритм LZ77 видає коди, що складаються з трьох елементів:

- зсув в словнику щодо його початку підрядка, співпадаючого з вмістом буфера;
- довжина підрядка;
- перший символ в буфері, наступний за підрядком.

Алгоритми групи LZ, мають істотний недолік. Оскільки вони використовують як словник тільки невеликий фрагмент повідомлення, то немає можливості кодувати

підрядки, що повторюються, відстань між якими в повідомленні більше, ніж розмір словника. Крім того, алгоритми обмежують розмір підрядка, який можна закодувати.

Чисто механічний підхід до поліпшення характеристик кодера LZ за рахунок збільшення розмірів словника і буфера звичайно дає зворотні бажаним результати. Припустімо, що ми збільшимо розмір словника до 64 К і розмір буфера до 1 К. Це приведе до того, що для кодування зсуву в словнику знадобиться 16 бітів замість 12, а для кодування довжини збігу буде потрібно 10 бітів замість 4. Таким чином кожна фраза буде закодована в 27 бітів замість 17, що, по-перше, приведе до значного зниження ступеня стиску на коротких повідомленнях, а по-друге, зробить неможливим кодувати підрядки, що повторюються та завдовжки менше 4 байтів.

Оскільки перераховані проблеми не могли не турбувати творців алгоритму Якоба Зіва і Абрама Лемпела, в 1978 році вони в роботі [19] запропонували нову версію свого алгоритму, яка далі називатиметься LZ78. LZ78 йде від ідеї ковзаючого по тексту вікна. На відміну від LZ77, в LZ78 словником є потенційно нескінченний список вже проглянутих ФРАЗ, а не підрядків, як в LZ77. У LZ78 і кодер, і декодер починають роботу з "майже порожнього"словника, що містить тільки один закодований рядок - НУЛЬ рядок. Коли кодер прочитує черговий символ повідомлення, символ додається до поточного рядка. Доти, доки поточний рядок відповідає якій-небудь фразі із словника, процес продовжується. Але, рано чи пізно, поточний рядок перестає відповідати якій-небудь фразі словника. У цей момент, коли поточний рядок є "останнім збігом"із словником, плюс тільки що лічений символ повідомлення, кодер LZ78 видає код, що складається з індексу збігу і наступного за ним символу, що порушив збіг рядків. Окрім того, нова фраза, що складається з індексу збігу і наступного за ним символу, додається в словник. Наступного разу, коли ця фраза з'явиться в повідомленні, вона може бути використана для побудови довшої фрази, що підвищує ступінь стиску інформації.

У 1984 році Terry A. Welch опублікував роботу [17], в якій приведена модифікація алгоритму LZ78, яка одержала назву LZW (Lempel-Ziv-Welch).

Алгоритм роботи кодера LZW можна описати таким чином:

1. Провести ініціалізацію словника односимвольними фразами, відповідними символам вхідного алфавіту (звично це 256 ASCII символів);
2. Прочитати перший символ повідомлення в поточну фразу PHRASE;
3. LABEL: Прочитати черговий символ повідомлення SYMBOL;
4. IF КІНЕЦЬ ПОВІДОМЛЕННЯ Видати код PHRASE;
5. EXIT;
6. END IF;
7. IF фраза PHRASE+SYMBOL вже є в словнику, Замінити PHRASE на код фрази PHRASE+SYMBOL;
8. GOTO LABEL;

9. ELSE Видати код PHRASE;
10. Додати PHRASE+SYMBOL в словник;
11. GOTO LABEL;
12. END IF;

Алгоритм декодування LZW може бути описаний таким чином:

1. CODE = Прочитати перший код повідомлення();
2. Попередній CODE = CODE;
3. Видати символ SYMBOL, у якого CODE(SYMBOL) == CODE;
4. LABEL: Наступний код: CODE = Прочитати черговий код повідомлення();
5. Вхідний CODE = CODE;
6. IF КІНЕЦЬ ПОВІДОМЛЕННЯ
7. EXIT;
8. END IF;
9. Наступний символ: Якщо CODE == CODE(PHRASE+SYMBOL) Видати SYMBOL;
10. CODE = CODE(PHRASE);
11. GOTO LABEL;
12. ELSE IF CODE == CODE(SYMBOL) Видати SYMBOL;
13. Додати в словник (Попередній CODE, SYMBOL);
14. Попередній CODE = Вхідний CODE;
15. GOTO LABEL;
16. END IF;

При декодуванні може скластися виняткова ситуація, коли кодер намагається закодувати повідомлення ABABA, де фраза AB вже присутня в словнику. Кодер виділить AB, видасть код(AB) і додасть ABA в словник. Потім він виділить ABA і відправить тільки що створений код (ABA). Декодер при отриманні коду (ABA) ще не додав цей код в словник, тому що ще не знає символ-розширення попередньої фрази. Проте, коли декодер зустрічає невідомий йому код, він може визначити, який символ видавати першим. Це символ-розширення попередньої фрази, який буде останнім символом поточної фрази, який був останнім символом попередньої фрази, який був останнім розкодованим символом. Для обробки цієї ситуації зручно скористатися стеком. Традиційно функцію поміщення в стек позначимо PUSH. а витягання із стека – POP.

Модифікований алгоритм декодування виглядає таким чином:

1. CODE = Прочитати перший код повідомлення();
2. Попередній CODE = CODE;
3. Видати символ SYMBOL, у якого CODE(SYMBOL)== CODE;
4. Останній SYMBOL = SYMBOL
5. Наступний код: CODE = Прочитати черговий код повідомлення();
6. Вхідний CODE = CODE;
7. IF КІНЕЦЬ ПОВІДОМЛЕННЯ EXIT;
8. END IF;
9. IF Невідомий(CODE) // Обробка виняткової ситуації
10. Видати(Останній SYMBOL)
    CODE = Попередній CODE
    Вхідний CODE = CODE (Попередній CODE, Останній SYMBOL)
11. END IF;
12. LABEL: Наступний символ:
    Якщо CODE == CODE(PHRASE+SYMBOL) PUSH(SYMBOL);
    CODE = CODE(PHRASE);
13. Повторити Наступний SYMBOL;
14. Інакше якщо CODE == CODE(SYMBOL) Видати SYMBOL;
15. Останній SYMBOL = SYMBOL;
16. Поки стек не порожній POP;
17. Кінець поки;
18. Додати в словник (Попередній CODE, SYMBOL);
19. Попередній CODE = Вхідний CODE;
20. GOTO LABEL;
21. END IF;

Очевидно, що декодер LZW використовує той же словник, що і кодер, будуючи його за аналогічними правилами при відновленні стиснутого повідомлення. Кожен прочитуваний код розбивається за допомогою словника на попередню фразу w і символ K. Потім рекурсія продовжується для попередньої фрази w до тих пір, поки вона не виявиться кодом одного символу, що і завершує декомпресію цього коду. Оновлення словника відбувається для кожного декодованого коду, окрім першого. Після завершення декодування коду його останній символ, сполучений з попередньою фразою, додається в словник. Нова фраза одержує те ж значення коду (позицію в

словнику), що привласнив їй кодер. Так, крок за кроком, декодер відновлює той словник, який побудував кодер.

Розглянемо приклад для трьохсимвольного алфавіту: A,B,C. Потік символів виглядає як ABACABABABA.

Кодування. Спочатку ми ініціалізуємо нашу таблицю ланцюжків: 0=A, 1=B, 2=C.

Перший символ є A, який входить в таблицю ланцюжків, отже PHRASE стає рівним A. Далі ми беремо AB, яка не входить в таблицю, отже ми виводимо код 0 (для PHRASE), і додаємо AB в таблицю ланцюжків з кодом 3. PHRASE стає рівним B. Далі ми беремо PHRASE+A = BA, яка не входить в таблицю ланцюжків, отже виводимо код 1, і додаємо BA в таблицю ланцюжків з кодом 4. PHRASE стає рівним A. Далі ми беремо AC, яка не входить в таблицю ланцюжків. Виводимо код 0, і додаємо AC в таблицю ланцюжків з кодом 5. Тепер PHRASE рівно C. Далі ми беремо PHRASE+A = CA, яка не входить в таблицю. Виводимо 2 для C, і додаємо CA до таблиці під кодом 6. Тепер PHRASE=A. Далі ми беремо AB, яка входить в таблицю ланцюжків, отже PHRASE стає рівним AB, і ми шукаємо ABA, якої немає в таблиці ланцюжків, тому ми виводимо код для AB, який дорівнює 3, і додаємо ABA в таблицю ланцюжків під кодом 7. Далі, PHRASE рівно A, ланцюжок AB вже закодований, тому PHRASE+=B, наступний ланцюжок – ABA також закодована, отже, як PHRASE вже беремо ABA і одержуємо PHRASE+B=ABAB, яке в таблиці кодів ще не зустрічалося, тому додаємо ABAB з кодом 8, а в кодову послідовність записуємо код ABA=7. На наступному кроці PHRASE рівно A. Ми не можемо більш узяти символів, тому ми виводимо код 0 для A і закінчуємо. Отже, потік кодів рівний 0,1,0,2,3,7,0.

Попередній код	Наявний символ	Наявна послідовність	Додається в словник	Кодова послідовність
	A	A		
A	B	AB	3	0
B	A	BA	4	1
A	C	AC	5	0
C	A	CA	6	2
A	B	AB		
AB	A	ABA	7	3
A	B	AB		
AB	A	ABA		
ABA	B	ABAB	8	7
B	A	BA		
BA		BA		0

Декодування.

Процес декодування практично повністю повторює процедуру кодування.

Також, як і при кодуванні, спочатку ініціалізуємо таблицю ланцюжків: 0=A, 1=B, 2=C.

Тепер перейдемо до процедури декодування початкової послідовності 0,1,0,2,3,7,0.

Перший код 0 відповідає символу A, другий код 1 відповідає символу B. Обидва коди входять в таблицю. Далі ми беремо ланцюжок 01, відповідно AB, яка не входить в таблицю, отже ми додаємо AB в таблицю ланцюжків з кодом 3. PHRASE стає рівним 1=B. Далі ми беремо PHRASE+0 = BA, яка не входить в таблицю ланцюжків, отже виводимо символ A, і додаємо BA в таблицю ланцюжків з кодом 4. PHRASE стає рівним 2=A. Далі ми беремо A2=AC, яка не входить в таблицю ланцюжків. Виводимо символ З, і додаємо AC в таблицю ланцюжків з кодом 5. Тепер PHRASE рівне 3=C. Далі ми беремо PHRASE+0 = CA, яка не входить в таблицю. Виводимо 2 З, і додаємо CA до таблиці під кодом 6. Наступний код 7.

Важливо!! Оскільки коду 7 в словнику немає, то конструюємо фрагмент для нього з поточного фрагмента+перший символ поточного фрагмента, одержуємо ABA і додаємо в таблицю ланцюжків під кодом 7. На наступному кроці PHRASE рівно A. Ми не можемо більш узяти символів, тому ми виводимо код 0 для A і закінчуємо. Таким чином початковий потік даних є ABACABABABA.

Вхідні дані	Вихідні дані	Додається в словник
0	A	
1	B	AB(3)
0	A	BA(4)
2	З	AC(5)
3	AB	CA(6)
7	ABA	ABA(7)
0	A	

Як вже згадувалося, Тері Велч працював над створенням програми compress разом з групою програмістів Unix. При реалізації compress він дещо удосконалив початковий алгоритм LZ78. Найцікавіше удосконалення - адаптивне скидання (adaptive reset). Замість того, щоб очищати словник після досягнення певного об'єму пам'яті для його зберігання (варіант, розглянутий до цього), програма compress продовжує використовувати словник до тих пір, поки залишається високим рівень стиску. Цей підхід базується на двох спостереженнях, що стосуються процесу стиску LZW. Одне з їх полягає у тому, що на перших порах величина стиску сильно залежить від розміру словника. Більший словник містить довші послідовності, які в результаті ефективно стискаються в одиночний код. Адаптивне скидання намагається експлуатувати великий словник якомога довше. Інше спостереження полягає у тому, що багато файлів (особливо архіви TAR (архіви системи UNIX), що містять різні види файлів усередині архіву) містять області з різними типами даних. В процесі роботи алгоритму LZW формується словник, спеціально пристосований для певного типу даних. При значній зміні типу даних словник вже не даватиме високого ступеня стиску. Контролюючи ефективність стиску, програма compress може скидати словник у момент падіння продуктивності.

При стисненні даних (надходженні на вхід програми чергової порції) програма на основі LZW намагається знайти в словнику фрагмент максимальної довжини,

співпадаючий з даними, замінює знайдену в словнику порцію даних кодом фрагмента і доповнює словник новим фрагментом. При заповненні всього словника (розмір словника обмежений за визначенням) програма очищає словник і починає процес заповнення словника знову. Реалізації цього методу розрізняються конструкцією словника, алгоритмами його заповнення і очищення при переповнюванні. Звичайно, при ініціалізації словник заповнюється початковими (елементарними) фрагментами – всіма можливими значеннями байта від 0 до 255. Це гарантує, що під час вступу на вхід чергової порції даних буде знайдений в словнику хоча б однобайтовий фрагмент. Алгоритм LZW резервує спеціальний код, який вставляється в упаковані дані, відзначаючи момент скидання словника. Значення цього коду звичайно приймають рівним 256. Таким чином при початку кодування мінімальна довжина коду складає 9 біт. Максимальна довжина коду залежить від об'єму словника – кількості різних фрагментів, які туди можна помістити. Якщо об'єм словника вимірювати в байтах (символах), то очевидно, що максимальна кількість фрагментів рівна числу символів, а, отже, максимальна довжина коду рівна $\log_2 W$ (W – об'єм словника в байтах).

2.3 Арифметичний стиск

Арифметичне стиснення, як і алгоритм Хаффмана використовує більш короткі коди для елементів, що часто з'являються, і довші для тих, що рідко з'являються, хоча подібно алгоритму LZW стискає послідовності величин, а не самі величини. Арифметичне стиснення достатньо близько лежить біля теоретичної межі стиску.

Арифметичне стиснення включає відображення кожної відмінної послідовності величин в діапазон цифр між 0 і 1. Потім ця область представляється як двійковий дріб змінної точності, при цьому менш загальні послідовності вимагають вищої точності (більше біт).

Розглянемо приклад. Розглянемо набір величин з частотою їх появи

$$a : 100 \quad b : 300 \quad c : 300 \quad d : 200 \quad e : 900 \quad f : 100$$

Таким чином вірогідність появи кожної з цих величин з точністю до четвертого знаку

$$P(a) = 0.0526, \quad P(b) = 0.1579, \quad P(c) = 0.1579,$$

$$P(d) = 0.1052, \quad P(e) = 0.4737, \quad P(f) = 0.0526.$$

Серія елементів ab матиме вірогідність $P(a) \times P(b)$, проте якщо поставити кожній серії у відповідність її вірогідність, це не забезпечить її унікальність, в даному випадку таку ж вірогідність має пари bf.

Представимо величину b сегментом рядка від 0.0526 до 0.2105. Довжина цього сегменту є вірогідність $P(b)$ появи елементу b. Величина c має ту ж вірогідність, але розташована в іншому діапазоні: від 0.2105 до 0.3684. Тепер можна однозначно представити елемент числом, вказуючим на сегмент цього елементу. Так число 0.25 указує на елемент c. Для уявлення послідовностей з двох елементів можна розділити кожний з цих сегментів. Наприклад, послідовності ba (з вірогідністю 0.0526) буде представлена першими 5.26% цієї області, величина bb (з вірогідністю 0.1579)

наступними 15.79% і так далі. Тоді елемент ba буде представимо будь-яким числом в діапазоні від 0.0526 до 0.0554.

Таким чином ми одержуємо схему однозначного кодування будь-якої послідовності величин. При цьому потрібна велика точність (більше біт) для маловірогідних величин. Такі послідовності представляють невеликий сегмент і навпаки, потрібно менше біт для послідовностей з високою вірогідністю. Чим більше сегмент, тим більше ймовірно знайти дріб низької точності, наприклад, 1/2, 1/4 або 1/8 для ідентифікації елементів цього сегменту.

Так само як і алгоритм Хаффмана, арифметичне кодування ефективне для тих, що часто повторюються послідовностей елементів. Ефективність алгоритму арифметичного стиснення може досягати 1:100.

Найважливішими властивостями арифметичного кодування є наступні:

- здатність кодування символу вірогідності p кількістю бітів довільно близьким до $-\log p$;
- вірогідність символів може бути на кожному кроці різною;
- дуже незначний запит пам'яті незалежно від кількості класів умов в моделі;
- велика швидкість.

У арифметичному кодуванні символ може відповідати дробовому кількості вихідних бітів. На практиці результат повинен, звичайно, бути цілим числом бітів, що відбудеться, якщо декілька послідовних високо вірогідних символів кодувати разом, поки у вихідний потік не можна буде додати 1 біт. Кожен закодований символ вимагає тільки одного цілочисельного множення і декількох додавань, для чого звичайно використовується тільки три 16-бітових внутрішніх регістра. Тому, арифметичне кодування ідеально підходить для адаптованих моделей і його відкриття породило безліч модифікацій, що набагато перевершує ті, що застосовуються разом з кодуванням Хаффмана.

Складність арифметичного кодування полягає у тому, що воно працює з накопичуваною вірогідністю розподілу, яка вимагає внесення для символів деякої впорядкованості. Відповідна символу накопичувана вірогідність є сума вірогідності всіх символів, передуючих йому.

2.4 Скалярне квантування

Розглядаючи задачі стиску зображень, ми не можемо не розглянути алгоритми квантування. Як правило, під квантуванням розуміють заміну неперервних даних системою індексів. В разі, коли дані описуються лише своїми значеннями, використовується скалярне квантування. Якщо під час квантування береться до уваги не тільки значення тих чи інших інформаційних характеристик, але і їхнє геометричне положення, то використовується векторне квантування.

У даному параграфі ми розглянемо деякі питання скалярного квантування.

Основні визначення

Хай $f = \{f_i\}_{i=0}^N$ масив вхідних даних. Надалі вважатимемо, що $\varphi = \{\varphi_i\}_{i=0}^N$ набір коефіцієнтів початкових даних, розгорнених у одновимірний масив. При цьому виходитимемо з того факту, що в розкладанні використовувалися ортогональні або близькі до ортогональних фільтри. В якості таких фільтрів можуть використовуватися фільтри на основі сплесків, наведених раніше, або перетворення Фур'є.

Для будь-якого масиву $\widetilde{\varphi} = \{\widetilde{\varphi}_i\}_{i=0}^N$ через $\widetilde{f} = \{\widetilde{f}_i\}_{i=0}^N$ позначимо результат застосування зворотного перетворення до $\widetilde{\varphi}$.

Через ортогональності фільтри має місце рівність Парсеваля

$$\sum_{i=0}^N f_i^2 = \sum_{i=0}^N \varphi_i^2, \qquad \sum_{i=0}^N \widetilde{f}_i^2 = \sum_{i=0}^N \widetilde{\varphi}_i^2$$

і, через лінійність процесу фільтрації і рівності Парсеваля маємо

$$\sum_{i=0}^N (f_i - \widetilde{f}_i)^2 = \sum_{i=0}^N (\varphi - \widetilde{\varphi}_i)^2. \tag{2.1}$$

Покладемо

$$A^- = \min_i \varphi_i, A^+ = \max_i \varphi_i,$$

і нехай, крім того, $B = B_{n+m+1}$ розбиття відрізка $[A^-, A^+]$ на $n+m+1$ проміжок точками

$$A_- = b_{-m-1/2} < b_{-m+1/2} < \ldots < b_{-1/2} < 0 < b_{1/2} < \ldots < b_{n+1/2} = A^+.$$

Для кожного індексу $i = 0, 1, 2, \ldots, N$ знайдеться індекс $k = k(i)$ $(k = -m, \ldots n)$ такий, що

$$b_{k-1/2} < \varphi_i \le b_{k+1/2}, \text{ якщо } \quad k = 1, \ldots, n, \tag{2.2}$$

або

$$b_{k-1/2} \le \varphi_i < b_{k+1/2}, \text{ якщо } \quad k = -1, \ldots, -m, \tag{2.3}$$

або

$$b_{-1/2} \le \varphi_i \le b_{1/2} \text{ якщо } \quad k = 0. \tag{2.4}$$

Легко бачити, що для будь-яких $\{a_i\}_{i=1}^M$ задача

$$\sum_{i=1}^M (a_i - c)^2 \to \min$$

має єдиний розв'язок

$$\hat{c} = \frac{1}{M} \sum_{i=1}^M a_i.$$

Таким чином якщо для фіксованого $k = -m, \ldots, n$ всі значення φ_i що знаходяться в одному з інтервалів (2.2), (2.3) або (2.4) замінені на одне і теж число c_k, то найменша похибка в середньоквадратичній метриці буде при

$$c_k = b_k,$$

де

$$b_k = \frac{s_k}{n_k}, \quad k = -m, \ldots, n$$

і

$$s_k = \sum_{i:k(i)=k} \varphi_i, \quad n_k = \sum_{i:k(i)=k} 1.$$

Тому послідовність

$$g_i = b_{(i)} (i = 0, 1, 2, \ldots, N)$$

буде якнайкращою (у значенні середньоквадратичного наближення) зі всіх послідовностей постійних на кожному з проміжків (2.3), (2.3) або (2.4).

Для фіксованих чисел b_k $(k = -m, \ldots, n)$(таблиця квантування) кожної послідовності g_i однозначно ставиться у відповідність послідовність

$$\psi_i = k(i).$$

Іноді замість цієї послідовності зручніше використовувати наступну послідовність

$$\theta_i = \begin{cases} 2k(i) - 1, & (i) > 0 \\ -2k(i), & k(i) < 0 \\ 0, & k(i) = 0. \end{cases}$$

Таким чином, при фіксованому векторі B кожне значення φ_i закруглено до значення g_i (значення ψ_i або θ_i називатимемо кодом масиву φ). Надалі цю процедуру називатимемо квантуванням послідовності φ по вектору B.

З (2.1) витікає, що похибка (RMSE (Root Mean Square Error)) виникла в результаті квантування

$$RMSE(B, f) = \|f - \widetilde{f}\|_{\ell_2},$$

де $\widetilde{f}$ результат застосування зворотного ортогонального (чи біортогонального) перетворення до масиву g, рівна

$$RMSE(B, f) = \|f - \widetilde{f}\|_{\ell_2} = \|\varphi - g\|_{\ell_2} = \sqrt{\frac{1}{N} \sum_{i=0}^{N} (\varphi_i - g_i)^2}.$$

Основною характеристикою якості відновлення є величина

$$p = PSNR(B, \varphi) = 20 \log_{10} \frac{\max_i |\varphi_i|}{RMSE(B, \varphi)}$$

PSNR (Peak Signal-to-Noise Ratio) – відношення сигнал/шум (у dB).

Процес квантування можна переписати в іншій термінології.

Покладемо

$$\Phi^+_{k+1/2} = \{\varphi_i : \varphi_i \le b_{k+1/2}\},$$

$$\Phi^-_{k+1/2} = \{\varphi_i : \varphi_i \ge b_{k+1/2}\}.$$

Крім того, хай для $k = 0, 1, 2, \ldots, n$

$$\Delta\Phi_k = \Phi^+_{k+1/2} \setminus \Phi^+_{k-1/2},$$

а для $k = -m \ldots, -1$

$$\Delta\Phi_k = \Phi^-_{k-1/2} \setminus \Phi^-_{k+1/2},$$

і

$$\Delta\Phi_0 = \Phi^+_{1/2} \bigcap \Phi^-_{-1/2}.$$

Тут $A \setminus B$ є теоретико-множинна різниця множин A і B.

Далі нехай n_k число елементів множини $\Delta\Phi_k$. Покладемо

$$b_k = \frac{1}{n_k} \sum_{i:\varphi_i \in \Delta\Phi_k} \varphi_i$$

тоді RMSE можна переписати у вигляді

$$RMSE(B, \varphi) = \sqrt{\frac{1}{N} \sum_{k=-m}^{n} \sum_{i:\varphi_i \in \Delta\Phi_k} (\varphi_i - b_k)^2}.$$

Цей процес легке алгоритмізувати, наприклад, для $k = 1, \ldots, n$ псевдокод виглядатиме таким чином

```
i=0;
for k=1 to n do
  begin
    s[k]=0; n[k]=0;
    if (fi[i]>b1[k])&(fi[i]<=b1[k+1]) then
       begin
        s[k]=s[k]+fi[i];
        n[k]=n[k]+1;
        psi[i]=k;
        i=i+1;
       else
        b[k]=s[k]/n[k];
    end if;
   end do.
```

Далі вважатимемо, що задані метод лінійного кодування послідовностей, що повторюються (RLE) і метод ентропійного кодування A. Як метод ентропійного кодування може бути метод Хаффмана, LZW, арифметичного кодування і ін.

Хай $V_0(\varphi)$ розмір (у байтах) початкового масиву $\{\varphi_i\}$ і $V_1(\varphi)$ розмір вихідного масиву, тобто розмір вихідних даних одержаних після застосування RLE і ентропійного кодування до послідовності θ_i або ψ_i.

Величину називають

$$c(B) = c(B, \varphi, RLE, A) = \frac{V_0(\varphi)}{V_1(\varphi)}$$

назвемо ступенем стиску даних (compression ratio або CR).

Таким чином для фіксованого масиву f (а отже, і масиву φ) кожен вектор B однозначно визначає точку на площині з координатами $(p(B,\varphi), c(B,\varphi))$. Ця точка характеризує якість квантування– ніж вона вища, тим квантуватель краще. Точну формулювання цього факту ми приведемо нижче.

Під правилом квантування $\mathfrak{M}$ розумітимемо процедуру побудови набору векторів $\{B\}$. Умовно розділятимемо цю процедуру на дві частини

- вибір нульового інтервалу квантування $[b_{-1/2}, b_{1/2}]$
- по вибраному нульовому інтервалу квантування побудова решти інтервалів $[b_{k-1/2}, b_{k+1/2}]$ $(|k| > 1)$.

Процедура процедура квантування полягає в наступному. Виберемо деяке число $\delta > 0$ і вважаємо

$$b_{1/2} = -b_{-1/2} = \delta.$$

Кількість інтервалів квантування n і m виберемо таким чином

$$n = [(A^+ - b_{1/2})\delta], \quad m = [(b_{-1/2} - A^-)\delta],$$

де $[\cdot]$ ціла частина числа і обчислимо межі інтервалів квантування

$$b_{k+1/2} = b_{1/2} + \frac{A^+ - b_{1/2}}{n}, \quad k = 1, \ldots, n$$

$$b_{k-1/2} = b_{-1/2} + \frac{b_{-1/2} - A^-}{m}, \quad k = -1, \ldots, -m$$

і таблицю квантування

$$b_k = \frac{1}{n_k} \sum_{i \in I_k} \varphi_i,$$

де

$$I_k = \{i : b_{i-1/2} \le \varphi_i < b_{i+1/2}\}$$

і n_k число елементів множини I_k.

Ця процедура дає набір B і називається рівномірним квантуванням. Міняючи δ одержуємо набір векторів B квантування, тобто одержуємо метод квантування.

Надалі вважатимемо, що фільтри, RLE, A і φ фіксовані. Метод квантування (вектор $\hat{B} = \hat{B}(p)$) називатимемо якнайкращим на класі W для заданої якості p_0, якщо

$$p(\hat{B}, p_0) \ge p_0$$

і для будь-якого іншого вектора $B \in W$ такого, що

$$p(B, p_0) \ge p_0$$

справедлива нерівність

$$(\hat{B}, p_0) \ge (B, p_0).$$

2.4.1 Вибір нульового інтервалу квантування

Розглянемо декілька підходів до рішення першої задачі – побудові нульового інтервалу квантування $[b_{-1/2}, b_{1/2}]$.

Позначимо через $\{\phi_i\}_{i=0}^N$ незростаючу перестановку послідовності $\{\varphi_i\}_{i=0}^N$, а через $\{\phi_i^+\}_{i=0}^N$ позначимо незростаючу перестановку множини $\{|\varphi_i|\}_{i=0}^N$. При заданій якості p_0 (PSNR) вважатимемо, що нульовий інтервал квантування дає, наприклад, 80% (це значення параметра підібране статистично) всієї погрішності. В цьому випадку число n_0 виберемо з умови

$$0.8\left(\phi_0^+ 10^{-p_0/20}\right)^2 \le \frac{1}{N-n_0}\sum_{i=0}^{N-n_0}\phi_i^+ \le 0.81\left(\phi_0^+ 10^{-p_0/20}\right)^2.$$

Розглянемо інший підхід до рішення цієї задачі.

Хай i_{-1} і i_1 є рішення екстремальної задачі

$$\begin{cases} \sum_{i=i_{-1}}^{i_1}\phi_i \to \min \\ i_1 - i_{-1} = n_0. \end{cases}$$

Тоді вважаємо

$$b_{1/2} = \phi_{i_1},$$

і

$$b_{-1/2} = \phi_{i_{-1}}.$$

Для дослідження якнайкращих методів квантування зручно використовувати саме цей підхід. Міняючи n_0 і підбираючи $b_{\pm 1/2}$ можна одержати якнайкращий метод квантування в деякому діапазоні зміни p.

Нульовий інтервал квантування можна також визначити виходячи з рішення наступною екстремальною задачі.

Хай числа $\delta_{-1}, \delta_1 > 0$, $\delta_{-1} + \delta_1 = \delta$. Через I_0 позначимо множину, що складається з n_0 індексів таких, що $-\delta_{-1} \le \varphi_i \le \delta_1$, тобто

$$I_0 = \{i : -\delta_{-1} \le \varphi_i \le \delta_1\}$$

і покладемо

$$\epsilon(\delta_{-1}, \delta_1) = \sum_{i\in I_0}\left(\varphi_i - \frac{1}{n_0}\sum_{i\in I_0}\varphi_i\right)^2.$$

Для будь-якого фіксованого $\varepsilon > 0$ через n_0 позначимо розв'язок екстремальної задачі

$$n_0 \to \max$$

при умові

$$\epsilon(\delta_{-1}, \delta_1) < \varepsilon.$$

Кожний з розглянутих підходів має свої достоїнства і недоліки. Зокрема, перший підхід використовує незростаючу перестановку модулів коефіцієнтів, другий, перестановку самих коефіцієнтів, для третього підходу перестановка не потрібна, проте

обчислювальна складність цього алгоритму не менше. Крім того, в цьому випадку потрібно достатньо точно запам'ятовувати величину

$$\frac{1}{n_0}\sum_{i\in I_0}\varphi_i$$

до якої квантуються значення в цьому інтервалі.

Перейдемо до розгляду другої задачі. При її рішенні виходитимемо з того факту, що числа $b_{-1/2}$ і $b_{1/2}$ вже вибрані.

2.4.2 Вибір інтервалів квантування

Раніше було розглянуте рівномірне квантування. Опишемо ще декілька методів квантування.

Розглянемо інший підхід до рішення другої задачі.

Вважатимемо, що задані межі інтервалу $[b_{1/2}, b_{-1/2}] = [\varphi_{i_1}, \varphi_{i_{-1}}]$. Обчислимо

$$i_2 = \left[\frac{i_1}{2}\right].$$

Хай α деякий параметр (на практиці його доцільно брати $\alpha \in [1/2, 2/3]$).

Якщо

$$\frac{1}{i_2^\alpha}\sum_{k=0}^{i_2}\left(\varphi_i - \frac{1}{i_2}\sum_{k=0}^{i_2}\varphi_i\right)^2 > \frac{1}{(i_1-i_2)^\alpha}\sum_{k=i_2}^{i_1}\left(\varphi_i - \frac{1}{i_1-i_2}\sum_{k=i_2}^{i_1}\varphi_i\right)^2,$$

то покладемо

$$i_3 = \left[\frac{i_2}{2}\right],$$

інакше

$$i_3 = i_2 + \left[\frac{i_1-i_2}{2}\right],$$

тобто проміжок з найбільшим значенням RMSE розбиваємо на дві рівні частини.

Далі цей процес продовжуємо аналогічним чином: вибираємо проміжок з найбільшим значенням RMSE і розбиваємо його на дві рівні частини. Цей процес продовжуємо або задане число раз або, використовуючи деякий критерій, наприклад, поки не одержимо необхідну величину похибки квантування.

Цей метод квантування можна модернізувати, вибираючи на кожному кроці i_k так, щоб сума значень RMSE на даному проміжку була мінімальною.

Далі розглянемо один метод квантування, який спирається на наступну лему.

Лема 1 *Хай $\phi(x)$ неперервна незростаюча функція, що задана на проміжку $x \in [0, T]$. Задача*

$$F(B) = \sum_{k=-m}^{n}\left(\int_{x_{k-1/2}}^{x_{k+1/2}}\left(\phi(x) - \frac{1}{x_{k+1/2}-x_{k-1/2}}\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz\right)^2 dx\right) \to \min$$

за умов

$$0 = x_{1/2} < \ldots < x_{n+1/2} = T \tag{2.5}$$

має єдиний розв'язок $\hat{x}_{k+1/2}$*, який визначається співвідношенням*

$$\phi(\hat{x}_{k+1/2}) = \frac{1}{2}\left(\frac{1}{\hat{x}_{k+1/2} - \hat{x}_{k-1/2}}\int_{\hat{x}_{k-1/2}}^{\hat{x}_{k+1/2}}\phi(x)dx + \right.$$

$$\left. + \frac{1}{\hat{x}_{k+3/2} - \hat{x}_{k+1/2}}\int_{\hat{x}_{k+1/2}}^{\hat{x}_{k+3/2}}\phi(x)dx\right), \quad k = 1, \ldots, n-1.$$

Для вирішення цієї екстремальної задачі використовуємо традиційні методи математичного аналізу.

Дійсно, для $k = 1, \ldots, n-1$

$$\frac{\partial F}{\partial x_{k+1/2}} =$$

$$= \int_{x_{k-1/2}}^{x_{k+1/2}} 2\left(\phi(z) - \frac{\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz}{x_{k+1/2} - x_{k-1/2}}\right)\left(\frac{\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz}{\left(x_{k+1/2} - x_{k-1/2}\right)^2} - \frac{\phi(x_{k+1/2})}{x_{k+1/2} - x_{k-1/2}}\right)dz +$$

$$+ \left(\phi(x_{k+1/2}) - \frac{\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz}{x_{k+1/2} - x_{k-1/2}}\right)^2 +$$

$$+ \int_{x_{k+1/2}}^{x_{k+3/2}} 2\left(\phi(z) - \frac{\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(z)dz}{x_{k+3/2} - x_{k+1/2}}\right)\left(-\frac{\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(z)dz}{\left(x_{k+3/2} - x_{k+1/2}\right)^2} + \frac{\phi(x_{k+1/2})}{x_{k+3/2} - x_{k+1/2}}\right)dz -$$

$$- \left(\phi(x_{k+1/2}) - \frac{\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(z)dz}{x_{k+3/2} - x_{k+1/2}}\right)^2 =$$

$$= \left(\frac{1}{x_{k+3/2} - x_{k+1/2}}\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(z)dz + \frac{1}{x_{k+1/2} - x_{k-1/2}}\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz\right) \times$$

$$\times \left(2\phi(x_{k+1/2}) - \frac{1}{x_{k+3/2} - x_{k+1/2}}\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(z)dz - \frac{1}{x_{k+1/2} - x_{k-1/2}}\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(z)dz\right).$$

Звідси прирівнюючи похідну нулю, відразу одержуємо

$$b_{k+1/2} = \phi(x_{k+1/2}) = \frac{1}{2}\left(\frac{1}{x_{k+1/2} - x_{k-1/2}}\int_{x_{k-1/2}}^{x_{k+1/2}}\phi(x)dx + \right.$$

$$\left. + \frac{1}{x_{k+3/2} - x_{k+1/2}}\int_{x_{k+1/2}}^{x_{k+3/2}}\phi(x)dx\right)$$

Приведений доказ не є строгим – необхідно показати що при цьому досягається мінімум і що одержана умова є не тільки необхідним, але і достатнім. Для цього потрібно детально розглянути випадок "злиплих"вузлів (коли деякі з нерівностей (2.5)) обертаються в рівність. Строгий доказ цього факту є громіздким і ми його наводити не будемо.

У декілька іншому вигляді цей результат приведений в роботах Ллойда і Макса (див. [4]).

Переписуючи затвердження леми в термінах приведених вище, приходимо до висновку, що при фіксованих $b_{1/2}$ $b_{-1/2}$ і числах n і m вектор B буде оптимальним по RMSE, тоді і тільки тоді, коли виконується рівність

$$b_{k+1} = 2b_{k+1/2} - b_k, \quad (k = -m, \dots, n,\ k \neq 0)$$

або, що те ж

$$b_{k+1} = b_{k+1/2} + (b_{k+1/2} - b_k), \quad \text{якщо} \quad k = 1, \dots, n, \tag{2.6}$$

і

$$b_{k-1} = b_{k-1/2} + (b_{k-1/2} - b_k), \quad \text{якщо} \quad k = -m, \dots, -1. \tag{2.7}$$

Рівність (2.6), (2.7) можна використовувати як у разі, коли задане число інтервалів квантування (тобто задані числа n, m) і вибирати b_k по рекурентним співвідношенням (2.6), (2.7), так і при заданих межах нульового інтервалу квантування $b_{-1/2}$, $b_{1/2}$ і значеннях b_{-1}, b_1. В цьому випадку при використовуванні співвідношень (2.8), (2.9), легко знайти решту інтервалів $[b_{k-1/2}, b_{k+1/2}]$, покласти $[b_{n-1/2}, b_{n+1/2}] = [b_{n-1/2}, A^+]$ і $[b_{m-1/2}, b_{m+1/2}] = [A^-, b_{m+1/2}]$. При цьому однозначно визначаються безперервні зліва функції $n(\delta)$ і $m(\delta)$. Для лівих меж в точках розривів цих функцій рівності (2.6), (2.7) виконуються для всіх інтервалів $[b_{k-1/2}, b_{k+1/2}]$ $(k = -m, \dots, n,\ k \neq 0)$.

Модифікуємо цей алгоритм, розглянувши замість співвідношень (2.6), (2.7) рівності

$$b_{k+1} = b_{k+1/2} + \lambda_{k+1}(b_{k+1/2} - b_k), \quad \text{якщо} \quad k = 1, \dots, n, \tag{2.8}$$

$$b_{k-1} = b_{k-1/2} + \lambda_{k-1}(b_{k-1/2} - b_k), \quad \text{якщо} \quad k = -m, \dots, -1, \tag{2.9}$$

де числа λ_k вибираються із статистики або інших міркувань, про що буде сказано нижче.

Ясно, що при $\lambda_k = 1$ рівність (2.8) та (2.9) переходять в рівність (2.6) та (2.7).

Нехай, як і раніше, $\{\phi_i\}_{i=0}^N$ спадаюча перестановка скінченної послідовності $\{\varphi_i\}_{i=0}^N$, а також задані межі нульового інтервалу $b_{1/2} = \phi_{i_1}$, $b_{-1/2} = \phi_{i_{-1}}$ і $b_0 = 0$.

Зафіксуємо числа λ_k. Наприклад, $\lambda_1 = \lambda_{-1} = 1/2$ і $\lambda_k = 1$ для $|k| > 1$.

Хай, спочатку, $k > 0$.

З (2.8), при $k = 1$ (при цьому визначено λ_1), знаходимо b_1. Цьому числу відповідає індекс i_1 такий, що $b_1 = \varphi_{i_1}$. Для $i = i_1 - 1$ виконується умова

$$\sum_{\nu=i}^{i_1-1} (\phi_\nu - b_1) \leq 0.$$

Послідовно зменшуючи індекс i на одиницю, однозначно знайдеться такий індекс i_2, що або

$$\sum_{i=i_2}^{i_1-1} (\phi_i - b_1) \leq 0$$

і

$$\sum_{i=i_2-1}^{i_1-1} (\phi_i - b_1) > 0$$

або досягається значення $i = 0$.

У першому випадку покладемо $b_{3/2} = \varphi_{i_2}$ і $b_2 = b_{3/2} + \lambda_2(b_{3/2} - b_1)$, в другому покладемо $n = 2$ і обчислимо

$$b_n = \frac{1}{i_n} \sum_{i=0}^{i_n - 1} \phi_i, b_{n+1/2} = A^+.$$

Нехай вже обчислені $b_{1/2}, b_1, \ldots, b_k$, тоді однозначно знайдеться такий індекс i_{k+1}, що або

$$\sum_{i=i_{k+1}}^{i_k - 1} (\phi_i - b_k) \leq 0$$

і

$$\sum_{i=i_{k+1}-1}^{i_k - 1} (\phi_i - b_k) > 0$$

або досягається значення $i = 0$.

У першому випадку покладемо $b_{k+1/2} = \varphi_{i_{k+1}}$ і $b_{k+1} = b_{k+1/2} + \lambda_{k+1}(b_{k+1/2} - b_k)$, в другому покладемо $n = k + 1$ і обчислимо

$$b_n = \frac{1}{i_n} \sum_{i=0}^{i_n - 1} \phi_i, \quad b_{n+1/2} = A^+.$$

Обчислення закінчуються, коли буде досягнуто значення $i = 0$.

Псевдокод цієї процедури виглядає наступним чином

```
j=i[1];s=0; k=1; m=0;
 while j>0 do
   begin
     s=s+(fi[j]-b[k]);
     m=m+1;
     if s*(s+(fi[j-1]-b[k]))<=0
       then j=j-1
       else
        begin
          k=k+1;
          i[k]=j;
          b[k]=fi[j]+lambda[k]*(fi[j]-b[k-1]);
          s=0;
          m=0;
        end;
   end if;
   n=k;
   b[n]=s/m;
  end do.
```

По аналогії з попередніми викладками, для $k < 0$ одержуємо індекс i_{k-1} з умов

$$\sum_{i=i_k+1}^{i_{k-1}} (\phi_i - b_k) \geq 0$$

і

$$\sum_{i=i_k+1}^{i_{k-1}+1} (\phi_i - b_k) < 0$$

поки $i_{k-1} + 1 < N$, інакше

$$b_k = \frac{1}{N - i_k - 1} \sum_{i=i_k+1}^{N} \phi_i, \quad b_{k-1/2} = A^-.$$

Псевдокод виглядає аналогічно.

Після того, як будуть визначені всі інтервали квантування $[b_{k-1/2}, b_{k+1/2}]$ потрібно перерахувати всі b_k, вважаючи

$$b_k = \frac{1}{i_k - i_{k+1} - 1} \sum_{i=i_{k+1}}^{i_k-1} \varphi_i.$$

Таким чином якщо числа λ_k фіксовані і вибирати $b_{1/2}$ і $b_{-1/2}$, наприклад, по числу n_0 елементів квантованих в нульовому інтервалі, то однозначно одержуємо набір інтервалів квантування $[b_{k-1/2}, b_{k+1/2}]$, тобто вектор B.

2.5 Кодування повторів (Run-Length Encoding)

Кодування повторів або метод групового стиску є одним із старих і найпростіших. На сучасному етапі він використовується в основному для стиску графічних файлів. Одним з перших поширених форматів, організованих за допомогою цього типу компресії, є формат PCX. "Класичний"варіант цього методу передбачає заміну послідовності символів, що повторюються, на рядок, що містить сам цей символ, і число (лічильник), яке показує, скільки разів цей символ повторюється. Наприклад, рядок 'AAAAAA CCCCCEEEFFFFFF' буде записано у вигляді: '6A5C3E6F'. При цьому можна перші чотири біти відвести під лічильник, другі під значення. Таким чином весь рядок буде записаний в чотирьох байтах.

Існує велика кількість різних модифікацій RLE. Всі вони так чи інакше пов'язані з структурою даних, які підлягають стиску. Перш за все структура методу групового стиску істотно залежить від використовуваного словника і вибору значень, до яких застосовується RLE. Як правило, використовування RLE в класичному значенні не тільки не зменшить загальний об'єм даних, але може навіть його збільшити. Тому, для ефективного використовування методу групового стиску треба використовувати ознаку повторів. Як правило, для ознаки повторів використовується прапорець, в якості якого може бути старший біт. Наприклад, якщо словник складається не більше ніж з 128 символів, то в байті, що визначає символ, сім біт відведені під його значення. Старший біт (прапорець) вказує на наявність чи відсутність повторів. Якщо

значення цього біта дорівнює нулю, то повторів немає, якщо одиниці, то наступний байт містить число повторів. В тому разі, якщо повторюючі ланцюжки істотно довше ніж 256, то старший біт лічильника можна також зарезервувати. Якщо старший біт лічильника дорівнює нулю, то решта семи біт містить значення цього лічильника, якщо одиниці, то під лічильник відводиться не тільки сім біт, що залишилися, але і наступний байт.

При стиску послідовностей елементів, що повторюються, доцільно використовувати ті закономірності, які є у послідовності даних. При використовуванні закономірностей зміни даних можна використовувати різні прийоми для того, щоб перегрупувати дані найвигіднішим для нас чином.

Наприклад, при використовуванні сплесків і застосування квантування до частотних доменів, відквантовані коефіцієнти не тільки утворюють повторюючі ланцюжки, але і їх структура така, що довжини цих ланцюжків мають загальну тенденцію до збільшення або зменшення, тобто дані згруповані таким образом, що їх значення по модулю "майже"спадають, тобто даних, які порушують закон спадання на багато менше.

Описані алгоритм використовує той факт, що якщо з послідовності даних викинути нулі, то з однієї сторони, значення, що залишилися, утворюватимуть "майже"спадаючу послідовність (то є їх розрядність спадатиме), а з другого боку, кількість "пачок"нулів, тобто кількість згрупованих в однозв'язну множину нулів в первинній послідовності, буде таке що їх розрядність буде зростати (але не монотонно).

Приведемо докладний опис алгоритму на прикладі.

Дані	2	3	-5	0	11	31	0	0	-3	-9	0	0	0	0	1	-1	1	0	1
Прапорець	1	1	0		1	0			1	0					1	1	0		1
Знак	1	1	0		1	1			0	0					1	0	1		1
Числа	2	3	5		11	31			3	9					1	1	1		1
Нулі				1			2				4							1	

Перший рядок таблиці містить вхідні дані.

Другий рядок – послідовність прапорців. Значення прапорця дорівнює одиниці відповідає тому, що за даним числом йде число відмінне від нуля, якщо ж значення прапорця дорівнює нулю, це значить, що за даним числом йде послідовність нулів (може бути ця послідовність полягає тільки з одного нуля).

Третій рядок містить прапорці знаків відмінних від нуля. Кожному числу послідовності відмінному від нуля ставиться у відповідність прапорець із значенням рівним 1, якщо це число позитивне і 0, якщо воно негативне.

У четвертому рядку стоять модулі значущих (тобто відмінних від нуля) значень послідовності даних.

Останній рядок містить довжини послідовностей нулів, що повторюються.

На основі цієї таблиці побудуємо наступні бітові послідовності

У зв'язку з тим, що в послідовності значущих чисел даних найбільшу питому вагу мають одиниці, інвертуємо значення останнього біта даних. В результаті одержимо наступну таблицю.

Набуті значення упорядкуємо по стовпцях – на початку перший стовпець (значення старшого біта), потім другий і так далі. Одержану послідовність групуємо від-

Число					Прапорець	Знак
0	0	0	1	0	1	1
0	0	0	1	1	1	1
0	0	1	0	1	0	0
	1	0	1	1	1	1
	1	1	1	1	0	1
0	0	0	1	1	1	0
	1	0	0	1	0	0
0	0	0	0	1	1	1
	0	0	0	1	1	0
	0	0	0	1	0	1
0	0	0	0	1	1	1

Число					Прапорець	Знак
0	0	0	1	1	1	1
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
1	1	1	1	0	0	1
0	0	0	1	0	1	0
0	1	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	1	1

водячи на одне число 4 біти і до одержаній послідовності застосовуємо традиційний метод RLE.

Значення кількості нулів, що повторюються, теж запишемо в таблицю

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	0	0	1

Також як і раніше, набуті значення упорядкуємо по стовпцях – на початку перший стовпець (значення старшого біта), потім другий і так далі. Одержану послідовність групуємо відводячи на одне число 4 біти і до одержаної послідовності застосовуємо традиційний метод RLE.

Розділ 3

Методи зберігання графічних даних.

Проглядаючи зміст жорсткого диска типового ПК, майже напевно знайдете багато файлів з розширеннями BMP, ICO, PNG, GIF, TIF, PCX і JPG. Всі ці файли містять растрові графічні зображення. Розширення в імені файлу говорить про те, в якому форматі зберігається інформація. Наприклад, розширення BMP позначає BMP-файл, підтримуваний в системах Windows і OS/2 (BMP - скорочення від bitmap, тобто бітовий, растровий); TIF - скорочення від TIFF або Tagged Image File Format.Это тільки два представники великого сімейства форматів, використовуваних на персональних комп'ютерах. Кожний з цих форматів по різному зберігає графічну інформацію, і кожний з них розроблявся під конкретні цілі. Формат GIF (Graphics Interchange File - файл графічного обміну), наприклад, був придуманий для того, щоб умістити якомога більше інформацію в обмеженому просторі з метою зменшити час отримання файлів для користувачів мережі CompuServe. Формат PCX спочатку був придуманий для зберігання чорно-білих графічних файлів, створюваних ранньою версією програми розфарбовування PC Paintbrush для комп'ютерів IBM PC.

Що містить растровий графічний файл? Такий файл звичайно містить інформацію двох видів: графічну і не графічну. У графічних даних указуються кольори пікселів, нерафічні дані містять іншу інформацію, необхідну для відновлення зображення, наприклад його висоту і ширину. (Якщо зображення містить 1 мільйон пікселів, то графічній програмі знати її розміри: чи малювати їй зображення 500 на 2000 або 1000 на 1000 пікселів?) Нерафічна частина файлу може також включати іншу інформацію, таку як номер версії або інформацію про авторські права. Все залежить від формату і від того хто (або який програмний пакет) створив цей файл. У растрових файлах використовується звичайно один з двох методів зберігання даних про пікселах. У повнокольорових зображеннях піксел може приймати будь-яке з більш, ніж 16 мільйонів значень, тому і колір піксела зберігається звичайно як 24-розрядне значення - по 8 бітів на червоний, зелений і синій компоненти кольору. Якщо зображення містить 1 мільйон пікселів, то розмір файлу буде рівний 3 мільйонам байтів плюс довжина нерафічних даних. Якщо ж зображення обмежено 256 або менш кольорами, то колірна інформація звичайно кодується з використовуванням палітри. Замість того, щоб зберігати значення кольору піксела, інформація про піксел указує на рядок в палітрі, а вона, у свою чергу, містить колір.

Розглянемо докладніше один з форматів – BMP, який широко використовується

в системах Windows.

3.1 Формат BMP

Перші 14 байтів BMP-файлу складають його заголовок.

Заголовок файлу містить три значення: букви BM (сігнатура), які говорять про те, що графічний файл має BMP формат, число, що означає розмір файлу і число, яке вказує на те, де знаходяться растрові дані. Останнє число дорівнює кількості байтів від початку файлу. Ще два поля в заголовку файлу зарезервовані для майбутніх потреб і звичайно містять нулі.

Необхідні неграфічні дані заховані в інформаційному заголовку. Поля в інформаційному заголовку, в числі інших, містять його розмір (40 байтів в BMP-файлах для Windows), висоту і ширину зображення в пікселах, і кількість бітів на піксел.

Таблиця кольорів (палітра) містить 256 полів по 4 байти. Перший байт в кожному полі відповідає за синю компоненту кольору, другий за зелену і третій - за червону. Четвертий байт не використовується і встановлюється в 0. Якщо перші три значення в таблиці кольорів дорівнюють 0, 192 і 192, то це значить, що нульовий номер відповідає жовтому кольору середньої інтенсивності (суміш зеленого і червоного). У палітрі визначені всі кольори, що використовуються в зображенні.

Решта частини файлу містить значення пікселів. Якщо зображення складається з 1 мільйона пікселів, а кожен піксел вимагає 8 бітів - одного байта колірної інформації, таким чином, ця частина файлу займає 1 мільйон байтів. Послідовність байтів відповідає порядку пікселів в зображенні: зліва направо, починаючи з нижнього рядка зображення. Значення кожного байта є номер кольору в таблиці кольорів.

Відображення на екран малюнка, що зберігається в BMP-файлі, починається з читання заголовка файлу і інформаційного заголовка. Програма таким чином отримує розміри зображення і кількість кольорів. Потім програма читає таблицю кольорів. Якщо комп'ютер виводить не більше 256 кольорів, то програма заповнює колірну палітру значеннями з таблиці кольорів. Таким чином, забезпечується правильна передача кольорів малюнка. Якщо використовуємо повнокольорову картинку - 24 або 32 біти на піксел, то колірну палітру заповнювати не потрібно. В цьому випадку на кожен піксел відводиться три (або чотири) байти значення, що містять синього, зеленого і червоного кольорів (для 32 бітного зображення ще один байт відводиться для альфа-компоненти, тоб-то ступеня прозорості даного піксела). Зауважимо, що кожен рядок доповнюється нулями до межі в 4 байта.

І, нарешті, прочитуються растрові дані. Як тільки рядок із значеннями пікселів прочитаний з файлу, він передається у відео-буфер для отримання зображення на екрані. У таких графічних середовищах як Windows, програма пересилає значення кольорів не безпосередньо у відео-буфер, а в Windows, і вже Windows заносить їх у відео-буфер. Зауважимо, що малюнок на екрані заповнюється не зверху вниз, а навпаки.

Опис формату bmp - файлу (24, 32 bits per pixel).

```
FILE HEADER   (Заголовок файлу)
```

```
BM signature (2 bytes) =BM
File size (4 bytes) Reserved (2 bytes) =0
Reserved (2 bytes)=0
Location of bitmap data (4 bytes)

INFORMATION HEADER  (Заголовок Bitmap)
Size of information header (4 bytes) =40
(Розмір заголовка) Image width (4 bytes) (Ширина зображення в пікселах)
Image height (4 bytes) (Висота зображення в пікселах)
Number of color planes (2 bytes) =1 (Число площин кольорів)
Number of bits per pixel (2 bytes) (Число бітів на піксел)
Compression method used (4 bytes) (Тип стиску)
Number of bytes of bitmap data (4 bytes) (Розмір стислого зображення або 0)
Horizontal screen resolution (4 bytes) (Горизонтальне розрішення  в пікс/метр)
Vertical screen resolution (4 bytes)(Вертикальне розрішення в пікс/метр)
Number of colors used in the image (4 bytes) =0 (Кількість кольорів)
Number of important colors (4 bytes) (Кількість важливих кольорів)
```

TABLE (Таблиця кольорів)

Палітра (таблиця кольорів) містить опис 256 відтінків кольорів по 4 байти на кожен.

BITMAP DATA. Дані описують колір пікселів що йдуть зліва направо і до верху знизу. Якщо зображення має глибину кольору 24 біти на піксел, то кожні три байти відповідають одному пікселу, визначаючи колірну гамму в послідовності BGR. Якщо глибина кольору 32 біти на піксел, то на кожну крапку відводиться ще один байт, що містить значення α- компоненти (прозорість).

Кожен рядок доповнюється нулями до кінця параграфа.

3.2 Конструкція формату JPEG.

У середині восьмидесятих років для вирішення задачі стиску повнокольорових зображень був розроблений формат JPEG (Joint Photographic Experts Group). Схема роботи методів стиску, що лежать в основі JPEG полягає в наступному. Після переводу повнокольорового зображення з колірного простору RGB (red, green, blue) в колірний простір YCrCb (Y- люмінесцентна складова, Cr- хроматичний червоний, тобто теплі тони і Cb- хроматичний синій- холодні тони), зображення розбивається на квадрати розміру 8×8 пікселів. До кожного з цих квадратів застосовується дискретне косинус- перетворення Фур'є. Для цієї мети створюється матриця D дискретного косинус-перетворення Фур'є, значення якої обчислюються за правилом

$$D_{i,j} = \frac{1}{n^2},$$

при $i = 0$ чи $j = 0$ та

$$D_{i,j} = \frac{4}{n^2} \cos \frac{\pi(2i+1)}{2n} \cos \frac{\pi(2j+1)}{2n},$$

для $i, j > 0$. Тут $i, j = 0, 1, ..., 7$ і $n = 8$.

Таким чином одержуємо матрицю D

$$\begin{pmatrix} .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 & .3536 \\ .4904 & .4159 & .2780 & .0979 & -.0971 & -.2773 & -.4154 & -.4903 \\ .4620 & 1916 & -.1909 & -.4617 & -.4623 & -.1924 & .1902 & .4614 \\ .4149 & -.0976 & -.4903 & -.2787 & .2767 & .4907 & .0995 & -.4145 \\ .3537 & -.3531 & -.3543 & .3526 & .3548 & -.3520 & -.3554 & .3514 \\ .2780 & -.4903 & .0963 & .4167 & -.4145 & -.1002 & .4910 & -.2747 \\ .1916 & -.4623 & .4614 & -.1894 & -.1938 & .4632 & -.4604 & .1872 \\ .0979 & -.2787 & .4167 & -.4909 & .4898 & -.4136 & .2740 & -.0924 \end{pmatrix}$$

Результатом застосування до матриці F первинних даних дискретного косинус - перетворення Фур'є, буде матриця

$$DF = D \times F \times D^T.$$

Для заданого коефіцієнта стиснення q обчислимо таблицю квантування Q за правилом

$$Q_{i,j} = 1 + (1 + i + j)q, i, j = 0, 1, ..., 7.$$

Процедура квантування полягає в обчисленні чисел

$$DFQ_{i,j} = \text{Round}\left(\frac{DF_{i,j}}{Q_{i,j}}\right),$$

де $Round(\cdot)$ є функція приведення до цілого.

Одержана матриця відквантованих коефіцієнтів містить велику кількість коефіцієнтів рівних нулю. Для послідовного ефективного використовування методу групового кодування, одержані дані упорядковуються відповідно до номерів матриці Z-зігзагу Jpeg

$$\begin{pmatrix} 1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\ 3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\ 4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\ 10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\ 11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\ 21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\ 22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\ 36 & 37 & 49 & 50 & 58 & 59 & 63 & 64 \end{pmatrix}$$

Унаслідок того факту, що людське око більш сприйнятливе до зміни яскравості (компоненти Y), ніж до зміни кольорів, при подальшій обробці колірних компонентів використовується децимація в пропорції 4:2:2 (береться тільки кожне четверте значення Cr і Cb) або 4:1:1 (береться тільки кожне восьме значення Cr і Cb). При реалізації зворотного ходу, значення яких бракує, знаходять використовуючи інтерполяційні формули.

До одержаних послідовностей застосовується 7-бітовий RLE кодування повторів (Run-Length Encoding). Подальша обробка полягає в подальшим застосуванні LZW з кодом змінної довгі і стисненні без втрат за допомогою коду Хаффмана.

Завдяки великій колекції варіантів кодування (таблиці квантування, z-обхід, різні методи групування коефіцієнтів Фур'є для різних колірних компонентів) цей формат достатньо ефективний. Проте для високих ступенів стиснення (більше 25 разів) стисле зображення візуально розпадається на квадратні ділянки постійності інтенсивності, що заважає нормальному сприйняттю образу. Крім того, з причини фіксованості таблиць квантування, одержати зображення із заданою ступенем стиску у форматі JPEG не завжди можливо.

Ці і інші супутні причини привели до необхідності розробки принципово нових методів стиску повнокольорових зображень.

Серед методів стиску, конкуруючих на право створення наступника JPEG, перш за все, розглядалися методи, засновані на фрактальному і на сплесковому (wavelets, або вейвлети) стиску даних.

Фрактальні методи засновані на виділенні і групуванні фрагментів зображення з схожою структурою. Основний складністю фрактальних методів є вибір геометричного примітиву, на якому порівнюється структура даних і можливість його обертання. Для реалізації фрактальних методів потрібні не тільки великі обчислювальні витрати, але і, найголовніше, вони вимагають досить великого часу.

Сплескові методи стиснення є продовженням і розвитком класичного аналізу Фур'є. У основі wavelet-методів стиснення лежить частотний аналіз характеристик зображення. При цьому, для низьких частот використовується акуратніша обробка, а для високих - грубіша.

Сплескові методи стиснення показують вищі показники, як за якістю стиснення, так і за часом роботи. Цей факт зумовив використовування саме сплескових методів для розробки наступника JPEG.

Численні дослідження в цьому напрямі вилилися, врешті-решт, в ухвалення нового стандарту - JPEG2000.

3.3 Особливості формату JPEG2000.

Розробка JPEG2000 почалася в 1996 році. Якраз до цього часу стала ясна істотна перевага алгоритмів стиснення на основі вейвлетів перед ДКП, використаним в JPEG. Проте, тільки ради збільшення ступеня стиску навряд стали б витрачати величезні кошти. При створенні нового стандарту, разом з досягненням більшої ефективності стиску ставилися ще і наступні цілі:

- Стійкість алгоритму до помилок каналу зв'язку при передачі стислого зображення. Тут видно націленість нового стандарту на мобільні застосування, на передачу зображень по радіоканалу.
- Уніфікована архітектура декодера. У JPEG 44 різних режимів декодування, залежно від його застосування. Синтаксис JPEG2000 такий, що в незалежності від вживаного способу кодування використовується один і той же декодер.
- Масштабованість. Залежно від потреби, це може бути масштабованість за розміром, дозволом. частотному змісту, кількості кольорів.

- Обробка окремих областей на зображенні (Region Of Interest). Наприклад, користувача може цікавити не все зображення вулиці, а лише фото окремої машини, він виділяє його (мишею), і декодер з високим якістю відновлює цей фрагмент. (Все неможливо "підняти"з високою якістю через обмеження на об'єм передаваної інформації).

- Стиск зображень великих розмірів.

- Можливість обробки стислого зображення без декомпресії.

Локальна, масштабована структура вейвлет-функцій дозволила вирішити вище перелічені задачі. Робочою групою були розглянуті сотні пропозицій дослідників і відібрані найперспективніші.

Основні блоки, що входять в структурну схему алгоритму стиску JPEG2000.

- Попередня обробка. Зображення, як правило, є набором ненегативних цілих чисел. На етапі попередньої обробки з нього віднімають середнє. Крім того, якщо зображення великого розміру, то воно може бути розбито на частини. Тоді кожна частина стискається окремо, а для запобігання появі помітних ліній на стику відновлених частин застосовуються спеціальні заходи.

- Вейвлет-перетворення. У JPEG2000 як фільтр використовується тензорний добуток одновимірних фільтрів. У першій частині стандарту визначені два вейвлет-фільтри - фільтр Добеши (9, 7) для стиснення з втратами і теж біортогональний фільтр з цілочисельними коефіцієнтами (5, 3) для стиснення без втрат. У другій частині стандарту дозволяється застосування будь-яких фільтрів, а також не тільки октавополосне розбиття, але і довільне (вейвлет-пакети і т.д.) В стандарті визначено, що вейвлет- перетворення здійснюється не шляхом згортки з імпульсними характеристиками фільтрів, а на основі алгоритму, відомого як ліфтінгова схема.

- Квантування. У першій частині стандарту визначено рівномірне квантування, у якого нульовий інтервал удвічі ширший за інших. У разі стиску без втрат розмір кроку квантування дорівнює 1, інакше він вибирається залежно від необхідного ступеня стиску.

- Ентропійне кодування. Застосовується адаптивний арифметичний кодер (а в JPEG був кодер Хаффмана). Зважаючи на патентні обмеження використовується не QM-кодер розробки IBM, а трохи гірший MQ-кодер, спеціально розроблений для JPEG2000.

- Кодування ведеться не всього зображення в цілому і навіть не окремих субсмуг, а дрібніших об'єктів - кодованих блоків (КБ). Розмір кодованого блоку може бути не більше 4096 пікселів, висота не менше 4 пікселів. Таке розбиття хоча і знижує декілька коефіцієнт стиску, але підвищує стійкість стислого потоку до похибок каналу зв'язку: похибка зіпсує лише невеликий блок. Кодування блоків ведеться в три етапи, бітовими площинами.

- Потік стиснутих даних упаковується в пакети.

Ліфтінгові схеми фільтрації, реалізовані у форматі JPEG2000.

У форматі JPEG2000 реалізовано два режима стиску зображень – без втрат і з втратами. Для стиску зображень без втрат використовується цілочисельне перетворення, яке має наступний вигляд –

Декомпозиція (прямий хід)

$$c^1_{2n} = c^0_{2n} - \left[\frac{c^0_{2n-1} + c^0_{2n+1} + 2}{4}\right],$$

$$c^1_{2n+1} = c^0_{2n+1} + \left[\frac{c^1_{2n} + c^1_{2n+2}}{2}\right].$$

Реконструкція (зворотний хід)

$$c^0_{2n+1} = c^1_{2n+1} - \left[\frac{c^1_{2n} + c^1_{2n+2}}{2}\right].$$

$$c^0_{2n} = c^1_{2n} - \left[\frac{c^0_{2n-1} + c^0_{2n+1} + 2}{4}\right],$$

Для високих ступенів стиснення JPEG2000 використовує біортогональну пару 7-9. В цьому випадку ліфтінгова схема для декомпозиції виглядає таким чином

$$\begin{cases} c^1_{2n} \leftarrow kc^0_{2n}, & \text{перший крок,} \\ c^1_{2n+1} \leftarrow -k^{-1}c^0_{2n+1}, & \text{другий крок,} \\ c^1_{2n} \leftarrow c^1_{2n} - \delta\left(c^1_{2n-1} + c^1_{2n+1}\right), & \text{третій крок,} \\ c^1_{2n+1} \leftarrow c^1_{2n+1} - \gamma\left(c^1_{2n} + c^1_{2n+2}\right), & \text{четвертий крок,} \\ c^1_{2n} \leftarrow c^1_{2n} - \beta\left(c^1_{2n-1} + c^1_{2n+1}\right), & \text{п'ятий крок,} \\ c^1_{2n+1} \leftarrow c^1_{2n+1} - \alpha\left(c^1_{2n} + c^1_{2n+2}\right), & \text{шостий крок,} \end{cases}$$

де

$$\alpha = -1.586134342, \beta = -0.052980118, \gamma = 0.882911075, \delta = 0.443506852,$$

і масштабуючий коефіцієнт $k = 1.230174105$.

Реконструкція (відновлення) сигналу виробляється таким чином

$$\begin{cases} c^0_{2n+1} \leftarrow c^1_{2n+1} + \alpha\left(c^1_{2n} + c^1_{2n+2}\right), & \text{перший крок} \\ c^0_{2n} \leftarrow c^1_{2n} + \beta\left(c^0_{2n-1} + c^0_{2n+1}\right), & \text{другий крок,} \\ c^0_{2n+1} \leftarrow c^0_{2n+1} + \gamma\left(c^0_{2n} + c^0_{2n+2}\right), & \text{третій крок,} \\ c^0_{2n} \leftarrow c^0_{2n} + \delta\left(c^0_{2n-1} + c^0_{2n+1}\right), & \text{четвертий крок,} \\ c^0_{2n+1} \leftarrow -k^{-1}c^0_{2n+1}, & \text{п'ятий крок,} \\ c^0_{2n} \leftarrow kc^0_{2n}, & \text{шостий крок.} \end{cases}$$

3.4 Сучасні методи кодування відеоінформації

Телевізійний кадр стандарту PAL містить 576 активних рядків (всього їх 625, але частина з них - службові). Згідно стандарту ITU-R BT.601 міжнародного телекомунікаційного співтовариства (ITU - International Telecommunications Union) кожен рядок містить 720 незалежних відліків. Таким чином, телевізійний кадр є матрицею з 720x576 точок, а гранично досяжний дозвіл обмежений 700 лініями. У оцифрованому телевізійному сигналі кожен кадр є точковим малюнком, де точка утворена відліком в горизонтальному рядку. Таких "малюнків"повинне проходити 25 за секунду (якщо строго - 50 напівкадрів полів, що складаються з парних і непарних рядків відповідно). Тоді інформаційний об'єм однієї хвилини цифрового відеосигналу з дозволом, відповідним мовному, і при глибині кольору 24 біти (True Color) складе 720x576 точок x 24 біта кольоровості x 25 кадрів/с x 60 с = 1866 Мб... Тобто без малого 2 гігабайти. При цьому швидкість цифрового відеопотоку буде рівна 250 Мбит/с. Навіть якщо поступитися якістю і розглядати удвічі гірший розподіл по обох вісях (360x288), що приблизно відповідає якості VHS-запису), об'єм хвилини відеопрограми займе 467 Мб, а відповідна швидкість цифрового потоку складе більше 60 Мбит/с. Треба врахувати, що кожен фільм має і звуковий супровід. Виходить, що такий сигнал є дуже громіздким для прямого використовування навіть в сучасних комунікаціях або на сучасних носіях.

Керуючись подібними орієнтирами, в рамках міжнародної організації по стандартизації (ISO) був створених комітет розробки міжнародних стандартів кодування і стиску відео- і аудіоінформації. Офіційне найменування цій групі було дане абсолютно невідтворне - ISO/IECJTC1 SC29 WG11. Згодом вона стала відома як "Експертна група по кінематографії"(Moving Picture Expert Group), а абревіатура MPEG, утворена від англійського варіанту повсякденної назви цієї групи, давно вже використовується як позначення розроблених нею норм і стандартів. У основу правил стиску відеоданих була закладена ідея пошуку і усунення надмірної інформації, що не впливає на кінцеве сприйняття якості зображення. В першу чергу, був врахований "людський чинник психофізіологічна модель сприйняття людиною відеозображень (HVS - Human Visual Sense); зокрема, той факт, що градації яскравості сприймаються зоровим апаратом людини значно тонше, ніж градації кольору. Це означає, що колірну інформацію можна зберігати "грубо"в порівнянні з яскравістю, при цьому в суб'єктивному сприйнятті якість зображення не погіршає. Тобто першочерговим напрямом в побудові алгоритмів всіх стандартів MPEG стає відшукання і усунення інформації, надмірної з погляду суб'єктивного сприйняття.

Працювала експертна група вельми плідно: за десятиліття розроблене ціле сімейство стандартів; більш того, майже всі вони живуть і успішно працюють. Кращим свідоцтвом тому служить той факт, що абревіатури MPEG і MP стали повсякденними на побутовому рівні. Розглянемо найважливіші етапи становлення MPEG.

MPEG1. Перший стандарт з'явився в 1992 р. і був розрахований на передачу відео по низькошвидкісних мережах або для запису на компакт-диски (Video-CD). Максимально можлива швидкість цифрового потоку була спочатку обмежена порогом в 150 кб/с (одношвидкісний CD-ROM або стандартний аудіопрогравач компакт-дисків). Щоб укладеться в задані рамки, звичайно, довелося поступитися якістю. У MPEG1

роздільна здатність картинки понижена, в порівнянні з розгорткою телебачення, в 2 рази по обох осях: 288 активних рядків в ТБ-кадрі і 360 відліків в активній частині ТБ-рядка. У принципі, ця якість близька по рівню до аналогового VHS-відеозапису. Але не можна забувати про JPEG-компресію. Зменшення числа відліків означає тим самим збільшення блоків і макроблоків усередині кожного кадру. Тобто зниження якості автоматично робить внутрішньокадрову компресію грубішою, і, як наслідок - помітнішої споживачу. Однотонні поверхні виявляються як би складені з квадратиків, що розсипаються; особливо настирливо квадратики "вилазять"на динамічних сценах.

Відомі випадки, коли при випуску версій фільмів на Video-CD доводилося урізувати у декілька разів багато сцен з великою кількістю руху: гонитва, бійки, вибухи і т.п. З цих причин, а також унаслідок прогресу цифрових технологій стандарт MPEG1 не встиг набути більшого поширення.

MPEG2. Час йшов, і прогрес у області цифрових технологій дозволив істотно удосконалити процес компресії відеоданих. Так з'явився новий стандарт MPEG2, робота над яким, власне, почалася відразу після виходу MPEG1 і завершилася в 1995 р. "Другий "MPEG не приніс революційних змін, це - цілком революційна доробка старого стандарту під нові можливості техніки і нові вимоги замовників - найбільших компаній mass-media. MPEG2 призначався для обробки відеозображення з телевізійною якістю, при пропускній спроможності каналів передачі даних від 3 до 15 Мбит/с. Зараз стандарт MPEG2 асоціюється у переважної більшості читачів і глядачів з DVD-дисками. Але в 1992 р., коли стартували роботи над цим стандартом, ще не існувало широкодоступних носіїв, на які можна було б записати відеоінформацію, стислу по алгоритмах MPEG2. Найголовніше - комп'ютерна техніка того часу не могла забезпечити і потрібну смугу пропускання. Зате супутникове телебачення з новітнім на ті часи устаткуванням вже тоді було готове надати канал передачі з необхідними характеристиками. У жовтні 1995 року через телевізійний супутник "Pan Am Sat"було реалізовано перше 20-канальне цифрове ТБ-віщання, що використало стандарт MPEG2. Супутник здійснював віщання на території Скандинавії, Бенілюксу, Близького Сходу і Північної Африки.

З появою ж у середині 90-х рр. цифрового багатоцільового диска DVD (Digital Versatile Disk, Digital Video Disk), що надає в простій - односторонній і одношаровій - версії місткість 4,7 Гб (майже в 8 разів більше CD), він, природно, стає практично безальтернативним масовим носієм для розповсюдження якісної продукції, стислої за стандартом MPEG2. Це зумовило масове виробництво бюджетних DVD-програвачів і, звичайно, появу недорогих апаратних кодерів/декодерів. На стандарті MPEG2 зараз побудовані всі системи цифрового супутникового телебачення. На ньому ж ґрунтуються ефірні системи цифрового телемовлення DVB, які одержують все більш широке розповсюдження у ряді країн Західної Європи і в США. У професійній студійній апаратурі для реалізації цифрового нелінійного монтажу використовується версія EDITABLE MPEG, в якій всі кадри ключові, а швидкість потоку у форматі 4:2:2 досягає 50 Мбит/с.

Революційних змін в новому стандарті немає, але удосконалення торкнулося практично всіх етапів "упаковки"; більш того, з'явилися операції, що раніше не засто-

совувалися. Наприклад, після розбиття відеопотоку на кадри і групи кадрів кодер аналізує зміст чергового кадру на предмет надмірних даних. Складається список оригінальних ділянок і таблиця ділянок, що повторюються. Оригінали зберігаються, копії видаляються, а таблиця ділянок, що повторюються, використовується при декодуванні стислого відеопотоку. Значне підвищення ступені стиску було досягнуте завдяки застосуванню у внутрішньокадровому стисненні нелінійного перетворювача Фурье замість лінійного. Оптимізації піддався алгоритм прогнозу руху, а також введені декілька нових алгоритмів компресії відеоданих. Вони в сукупності дозволяють кодувати різні шари кадру залежно від їх важливості з різною інтенсивністю цифрового потоку.

Стандарт MPEG2 надає можливість в процесі кодування задавати точність частотних коефіцієнтів матриці квантування, що безпосередньо впливає на якість і розмір одержуваного в результаті стиснення зображення. Точність квантування може варіюватися в діапазоні 8-11 біт на одне значення елементу. Для порівняння: у MPEG1 передбачалося тільки одне фіксоване значення - 8 біт на елемент. Тобто в рамках стандарту MPEG2 є можливість гнучкої настройки якості зображення залежно від пропускної спроможності мережі або місткості носія (от чому на перших DVD можна було бачити різне за якістю зображення). В той же час, користувачі таких апаратів, як DVD- або HD-рекордери, які використовують MPEG2-компрессию, знають, як можна самим задавати рівень якості запису (HQ, SP, LP і т.д.), міняючи таким чином обсяг записаного матеріалу. Ця гнучкість, зокрема, і зробила MPEG2 основою для прийому/передачі цифрового телебачення по різних цифрових мережах.

В результаті для фільмів, створених в стандартах PAL і SECAM, підтримується дозвіл 720x576 при 25 кадрах в секунду при високій якості. Власне, MPEG-фільм не можна віднести до якої-небудь системи кольорового телебачення, оскільки кадри в MPEG є просто картинками і не мають прямого відношення до початкової для фільму системи телебачення; може йтися про відповідність розміру і частоти проходження кадрів. У частині аудіо в MPEG2, в порівнянні з MPEG1, додана підтримка багатоканального звуку(Dolby Digital 5.1, DTS і т.п.)

MPEG3. Перш за все, не слід змішувати з широковідомим форматом компресії звуку MP3. Стандарт MPEG3 спочатку розроблявся для використовування в системах телебачення високої чіткості (High Definition Television, HDTV) із швидкістю потоку даних 20-40 Мбит/с. Але ще в процесі розробки стало ясно, що параметри, що вимагаються для передачі HDTV, цілком забезпечуються використовуванням стандарту MPEG2 при збільшеній швидкості цифрового потоку. Іншими словами, гострої потреби в існуванні окремого стандарту для HDTV немає. Таким чином, MPEG3, ще не народився як став складовою частиною стандарту MPEG2 і окремо тепер навіть не згадується.

MPEG4. У новому стандарті MPEG4, що з'явився у самому кінці 1999 р., запропонований ширший погляд на медіа-реальність. Стандарт задає принципи роботи з контентом (цифровим представленням медіа-даних) для трьох областей: власне інтерактивне мультимедіа (включаючи продукти, поширювані на оптичних дисках і через Інтернет), графічних додатків (синтетичного контента) і цифрового телебачення (DTV). Фактично даний стандарт задає правила організації середовища, причому

середовища об'єктний орієнтованою. Він має справу не просто з потоками і масивами медіа-даних, а з медіа-об'єктами (ключове поняття стандарту). Крім роботи з аудіо- і відеоданими, стандарт дозволяє працювати з природними і синтезованими комп'ютером 2D- і 3D-объектами, виробляти прив'язку їх взаємного розташування і синхронізацію один щодо одного, а також указує їх інтерактивну взаємодію з користувачем. Картинка розділяється на складові елементи - медіа-обьекти, описується структура цих об'єктів і їх взаємозв'язку, щоб потім зібрати їх в єдину відеозвукову сцену.

Такий спосіб представлення даних дозволяє змінити результуючу сцену, забезпечуючи високий рівень інтерактивності для кінцевого користувача і надаючи йому цілий ряд можливостей, наприклад: переміщати і поміщати об'єкти в будь-яке місце сцени, трансформувати об'єкти, змінювати їх форму і геометричні розміри, збирати з окремих об'єктів складовий об'єкт і виробляти над ним які-небудь операції, міняти текстуру і колір об'єкту, маніпулювати їм (примусити, наприклад, стіл пересуватися в просторі), міняти точку спостереження за всією сценою.

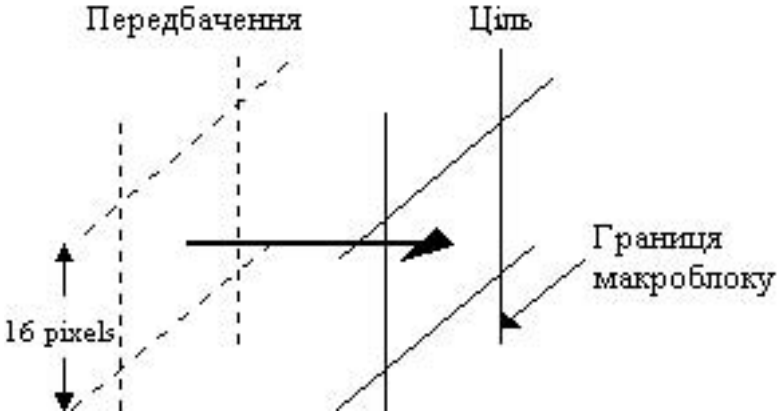
У боротьбі з конкурентами на ринку потокового відео (зокрема, згадаємо комп'ютерну компанію Apple і її QuickTime) в корпорації Microsoft зайнялися розробкою кодера, що дозволяє компресувати відеопотік відповідно до стандарту MPEG4. На одному з етапів відладки нового продукту бета-версія цього кодера стала надбанням широких мас... і хакерської громадськості. Отже, двоє хакерів, відомих під прізвиськами MaxMorice і Gej, презентували новий формат стиску відеофайлів, названий ними DivX ;-). Насправді це лише зламана версія Microsoft MPEG-4 Video Codec (Low-Motion кодек версія 4.1.00.4920 M$ MPEG4v3, а High-Motion кодек - 4.1.4917 M$ MPEGv3), як затверджують автори, вони прибрали глюки і трошки його поліпшили. Далі - більше: через приблизно півроку тепер уже цілком легальна фірма DivXNetworks Inc. переробила цей продукт і зняла з нього клеймо "Веселого Роджера". Microsoft ще на "піратській"стадії цієї історії по "політичних"мотивах звернула розробки в даному напрямі.

Особливу увагу надамо області стандарту MPEG4, яка нас найбільш цікавить - стиску відеоматеріалів. Алгоритм компресії відео, у принципі, працює за тією ж схемою, що і в попередніх стандартах, але є декілька радикальних нововведень.

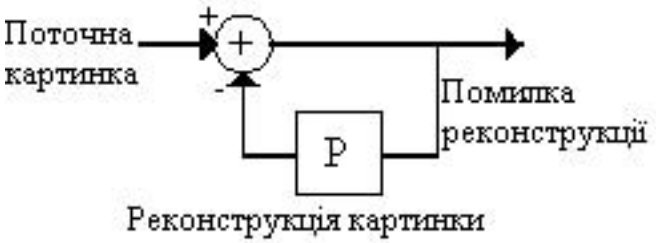
Основна ідея всієї схеми MPEG - це передбачати рух від кадру до кадру, а потім застосувати дискретне косинус перетворення (ДКП), щоб перерозподілити надмірність в просторі.

При обробці відеопослідовності кожне зображення може бути представлене у вигляді суперпозиції об'єктних відеоплощин (video object plane - VOP). Максимальна кількість таких площин - 256. Кожна площина також є зображенням у форматі YUV 4:2:0. Вона містить певну частину початкового (повного) зображення і, як правило, формально пов'язана з якоюсь візуально помітною його частиною. Площини кодуються незалежно. При декодуванні зображення "збирається"з окремих декодованих площин з використанням інформації, що зберігається окремо, про форму (shape) об'єктів, що містяться в тій або іншій площині (інформація про форму утворює так звані альфа-площини (alpha planes)). У окремому випадку ми маємо всього одну площину, яка містить все початкове зображення. Унаслідок відсутності ефективних методів

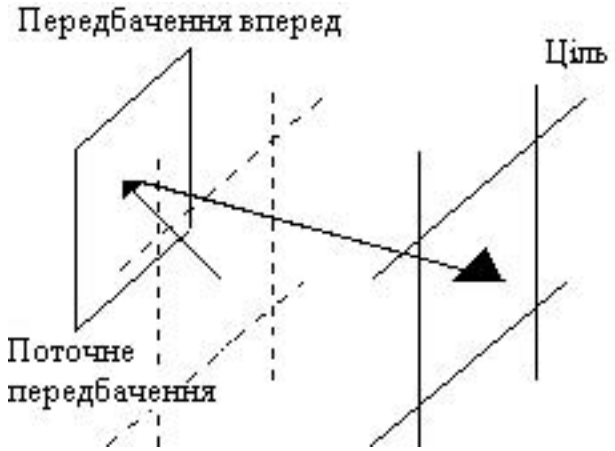
розподілу зображення на окремі об'єкти, саме цей випадок найчастіше зустрічається.



При обробці відеоплощини (фрейми) розділяються на чотири типи: I-площина, P-площина, B-площина і S-площина. I-площини кодуються без використання будь-якої інформації про інші площини. I-площину можна декодувати, не декодуючи ніякі інші площини, що дозволяє забезпечити швидкий доступ до довільних частин закодованого відеопотоку. Кожна I-площина розбивається на макроблоки розміру 16x16 (тут і далі указується розмір для компоненти Y; відповідний розмір для компонентів U і V - 8x8). Такі макроблоки називаються intra-макроблоками. Intra-макроблок у свою чергу розбивається на 4 intra-блоки розміру 8x8, кожен з яких бере участь в подальшому блоковому кодуванні/декодуванні (все сказане відноситься тільки до компоненти Y; для компонентів U і V виступає блок 8x8, який відповідає макроблоку компоненти Y). P-площини кодуються з використанням інформації про попередні I- і P-площини. P-площина також розбивається на макроблоки 16x16 і блоки 8x8, проте, крім intra-макроблоків і intra-блоків, тут присутні inter-макроблоки і inter-блоки. Перед кодуванням над inter-макроблоками/блоками здійснюється процедура компенсації руху (motion compensation): у попередніх площинах шукається макроблок/блок (він може знаходитися на довільній позиції, не обов'язково кратній 16 або 8), який максимально відповідає поточному inter-макроблоку/блоку (т.з. прогноз (prediction)), після чого знаходиться поточечна різниця; різницевий макроблок/блок (помилка прогнозу (prediction error)), що був отриманий в результаті цієї операції, надалі використовується при кодуванні. Важливо відзначити, що площини, на основі яких здійснюється процедура компенсації руху (опорні площини (reference planes)), повинні співпадати з площинами, які відновлюються в процесі декодування (при обчисленні помилки прогнозу у жодному випадку не повинна використовуватися інформація, доступна тільки на етапі кодування). Результатом дії описаної процедури крім блоків, що містять помилку прогнозу, є поле двокомпонентних векторів руху (motion vectors).

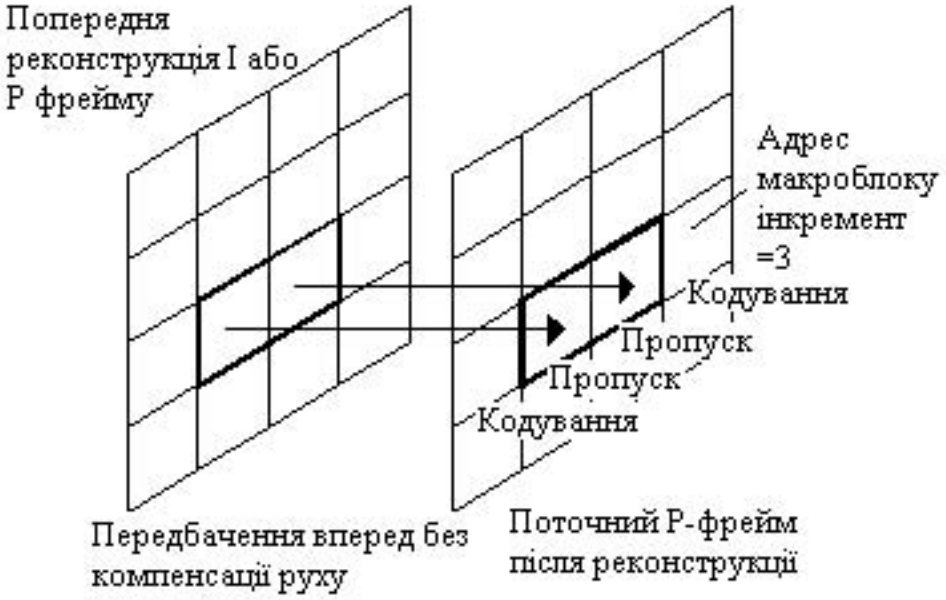


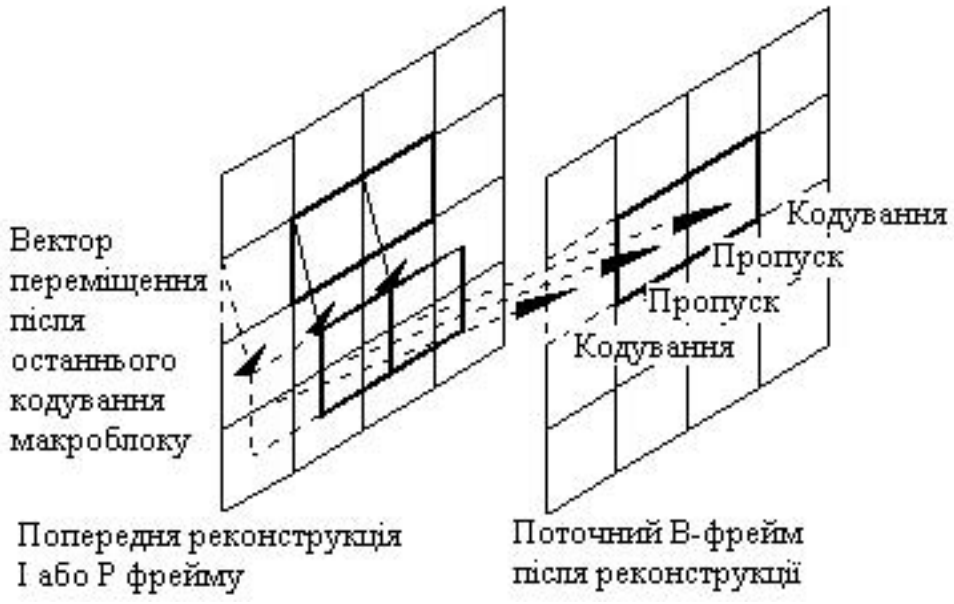
Пошук схожих блоків проводиться тільки для компоненти Y, вектори для компонентів U і V знаходяться шляхом ділення навпіл векторів для першої компоненти. Згідно стандарту, вектори руху мають в загальному випадку нецілу довжину, кратну 1/2 або 1/4. Для отримання нецілих векторів, опорні площини збільшуються (масштабуються) відповідно в 2 і в 4 рази по кожному з вимірювань (для збільшення використовується білінійна інтерполяція). Крім збільшення, стандарт визначає застосування так званого педдинга (padding). Процедура полягає в розширенні площини з усіх боків на величину, рівну 16 точками для компоненти Y і 8, - для компонентів U і V (і те і інше в цілих одиницях). Новим точкам привласнюються кольори найближчих до них граничних точок початкової площини. Педдінг (в даному випадку, можливо, має сенс використовувати термін доповнення/розширення) сприяє підвищенню ефективності компенсації руху поблизу меж відеоплощин. Компенсація руху може здійснюватися як макроблоками, так і блоками, тому для кожного макроблоку можуть бути знайдені або 1, або 4 вектори руху.



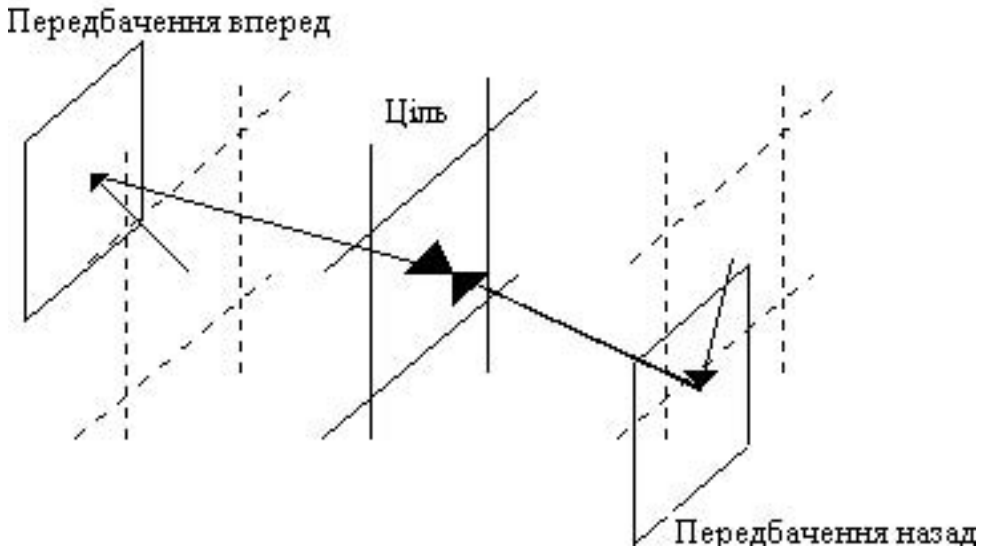
Вибір здійснюється на основі порівняння обсягів, які буде займати закодований макроблок в тому і в іншому випадку з урахуванням внеску коду вектора руху. Як було відмічено, P-площини можуть містити як inter-, так і intra-макроблоки. Завдяки компенсації руху, inter-макроблоки, як правило, кодуються з більшою ефективністю в порівнянні з intra-макроблоками, проте використання intra-макроблоків не пов'язане із зберіганням додаткової інформації про вектори руху, що знову ж таки може виявитися вигіднішим. Дилема вирішується так само, як і у попередньому випадку: тип використовуваного макроблоку вибирається виходячи з обсягу його сумарного коду. Стандарт передбачає особливий тип компенсації - компенсація з перекриттям

(overlapped motion compensation). Компенсація з перекриттям використовується тільки для блоків компоненти Y. При використанні цього методу різниця береться не між поточним блоком і деяким схожим блоком, який належнить попереднім площинам, а між поточним блоком і зваженою суперпозицією трьох схожих блоків. Відповідно вибираються наступні вектори: вектор для даного блоку і два вектори для блоків, сусідніх до даного блоку в макроблоці, який обробляється. B-площина є розширення поняття P-площини. Відмінність B-площини від P-площини полягає в тому, що при кодуванні B-площини для компенсації руху можуть використовуватися як попередні, так і подальші I- і P- площини (із зрозумілих причин, самі B-площини для цього використовуватися не можуть). Відповідно вводяться поняття прямого і зворотного прогнозу (forward and backward prediction). Кожен макроблок в B-площині може бути передбачений макроблоком на попередній площині, макроблоком на подальшій площині і суперпозицією цих макроблоків.





Використання двонаправленого прогнозу (bidirectional prediciton) дозволяє помітно підвищити ефективність прогнозу, а отже, і ефективність кодування.



Останній тип закодованої відеоплощини - S-площина - має безпосереднє відношення до спрайтів (sprite). Спрайтом в стандарті MPEG4 називається частина зображення, видима впродовж певного інтервалу у відеопослідовності. В ролі спрайту найчастіше виступає фон (задній план). При декодуванні з використанням спрайту фрейми частково відновлюються з окремих областей спрайту шляхом відображення цих областей на ту або іншу область декодованого фрейма з використанням перспективного перетворення (крім відображення, застосовуються також і інші спеціальні процедури). Спрайт зазвичай зберігається і кодується окремо, при цьому спосіб кодування ідентичний способу кодування I-площин. Окремим випадком використання

спрайту є S(GMC) - площини. В ролі спрайту тут виступає одна з попередніх I- або P-площин. Цей метод майже нічим не відрізняється від методу обробки P-площин, за тим виключенням, що процедура компенсації руху передує процедурі глобальної (вживаною до всієї площини) компенсації руху з використанням перспективного перетворення. Технологія GMC (global motion compensation), найчастіше застосовується для нівеляції руху і зміни оптичних параметрів камери. Площини різних типів, які ідуть поряд, розподіляються на відособлено кодовані групи. На початку групи повинна знаходитися I-площина, оскільки тільки вона може бути декодована першою. Далі розташовані площини інших типів. B- площини зазвичай чергуються з P- або S(GMC) - площини (хоча це і не обов'язково; наприклад, нерідко одна P- або S(GMC) - площини чередується з двома і найчастіше йдуть підряд B- площини). Розподіл площин на групи полегшує процес доступу до інформації, а також забезпечує велику перешкодостійкість. Останнє обумовлюється тим, що у разі помилки пошкодженим виявляється не весь закодований потік, а тільки окрема його частина.

Послідовність розкодованих кадрів звичайно виглядає наступним чином

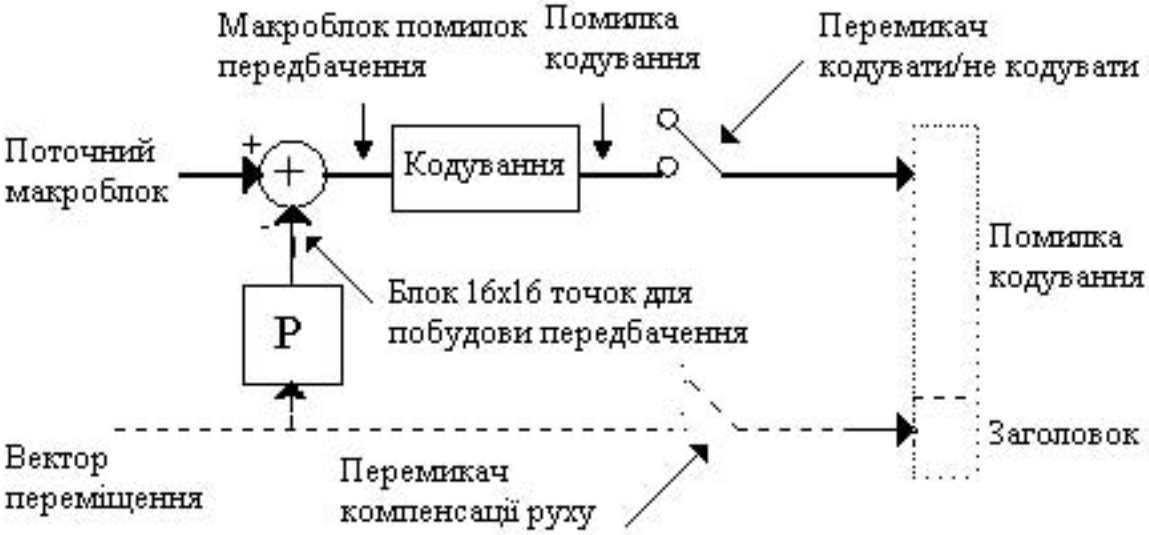
I B B P B B P B B P B B I B B P B B P B ...

Зауважимо, що від I до наступного I фрейму лежить 12 кадрів. Це засновано на вимозі довільного доступу, згідно якому початкова точка повинна повторюватися кожні 0.4 секунди. Співвідношення P і B засноване на досвіді. Щоб декодер міг працювати, необхідно, щоб перший P-фрейм в потоці зустрівся з першим B-фреймом, тому стислий потік виглядає наступним чином: 0 x x 3 1 2 6 4 5 ... де числа - це номери кадрів. xx може бути нічим, якщо це початок послідовності, або B-фрейми -2 і -1, якщо це фрагмент з середини потоку. Спочатку необхідно розкодувати I-фрейм, потім P, потім, маючи їх обидва в пам'яті, розкодувати B. Під час декодування P показується I-фрейм, B показуються відразу, а розкодований P показується під час декодування наступного. Прогноз макроблоків формується на основі відповідних 16x16 блоків точок (16x8 в MPEG-2) на попередніх відновлених кадрах. Ніяких обмежень на положення макроблоку в попередній картинці, окрім її меж, не існує.

Як тільки кадр розкодований, він стає не набором блоків, а звичним плоским цифровим зображенням з точок.

В MPEG розміри картинки, що відображається, і частота кадрів може відрізнятися від закодованого в потоці. Наприклад, перед кодуванням деяка підмножина кадрів в початковій послідовності може бути опущена, а потім кожен кадр фільтрується і обробляється. При відновленні кадру для створення початкового розміру і частоти кадрів використовується інтерполяція. Фактично, три фундаментальні фази (початкова частота, що кодована і показується) можуть відрізнятися в параметрах. Синтаксис MPEG описує кодовану частоту і частоту відновлених кадрів через заголовки, таким чином, початкова частота кадрів і розмір відомий тільки кодеру. Саме тому в заголовки MPEG-2 введені елементи, що описують розмір екрану для показу відеоряду. У I-фреймі макроблоки повинні бути закодовані як внутрішні - без посилань на попередні або подальші, якщо не використовуються режими масштабування. Проте, макроблоки в P-фреймі можуть бути як внутрішніми, так і мати посилання на попередні кадри. Макроблоки в B-фреймі можуть бути як внутрішніми, так і можуть посилатися на попередній кадр, подальший або на обидва. У заголовку кожного

макроблоку є елемент, що визначає його тип.



Послідовність кадрів може мати будь-яку структуру розміщення I, P і B фреймів. На практиці прийнято мати фіксовану послідовність (на зразок IBBPBBPBBPBBPBB), проте, сучасні кодери можуть оптимізувати вибір типу кадру залежно від контексту і глобальних характеристик відеоряду. Кожен тип кадру має свої плюси залежно від особливостей зображення (активність руху, тимчасові ефекти маскування та подібне). Наприклад, якщо послідовність зображень мало міняється від кадру до кадру, є сенс кодувати більше B-фреймів, ніж P. Оскільки B-фрейми не використовуються в подальшому процесі декодування, вони можуть бути стиснуті сильніше, без впливу на якість відеоряду в цілому. Вимоги конкретного застосування також впливають на вибір типу кадрів: ключові кадри, перемикання каналів, індексація програм, відновлення від помилок і т.д.

При стиску відео використовуються наступні методи:

1. Велика просторова кореляція зображення в цілому: прогноз першого низькочастотного коефіцієнта перетворення в блоці 8x8 (середнє значення всього блоку).
2. Статистика появи синтаксичних елементів в найвірогіднішому кодованому потоці: оптимальне кодування векторів руху, коефіцієнтів DCT, типів макроблоків і ін.
3. Розряджена матриця квантованих коефіцієнтів DCT: кодування нульових елементів, що повторюються, з позначенням кінця блоку.
4. Просторове маскування: ступінь квантування макроблоку.
5. Кодування ділянок з урахуванням змісту сцени: ступінь квантування макроблоку.
6. Адаптація до локальних характеристик зображення: кодування блоків, тип макроблоку, адаптивне квантування.

7. Постійний розмір кроку при адаптивному квантуванні: новий ступінь квантування встановлюється тільки спеціальним типом макроблоку і не передається за умовчанням.

8. Тимчасова надмірність: прямі і зворотні вектори руху на рівні макроблоків 16x16 точок.

9. Кодування помилки прогнозів макроблоків з урахуванням сприйняття: адаптивне квантування і квантування коефіцієнтів перетворення.

10. Тонке кодування помилки прогнозу на рівні макроблоків: кожний з блоків усередині макроблоку може бути кодований або пропущений.

11. Вектори руху - повільний рух фрагмента зображення з складним малюнком: прогноз векторів руху.

12. Появи і зникнення: прямий і зворотний прогноз в B-фреймах.

13. Точність міжкадрового прогнозу: білінійна інтерполяція різниці блоків.

14. Обмежена активність руху в P-фреймах: пропущені макроблоки. Коли вектор руху і помилка прогнозу нульові. Пропущені макроблоки дуже бажані в кодованому потоці, оскільки не займають бітів, окрім як в заголовку наступного макроблоку.

Розглянемо коротко процедури кодування intra/inter-блоков і векторів руху. Intra- та inter-блоки кодуються практично одним і тим же методом за невеликими виключеннями: спочатку проводиться блокове дискретне косинусне перетворення (discrete cosine transform - DCT), потім отримані в результаті перетворення коефіцієнти (DCT-коефіцієнти) квантуються, після чого квантовані коефіцієнти кодуються з використанням заздалегідь фіксованої системи кодів змінної довжини. Основна відмінність методу кодування intra-блоків від методу кодування inter-блоків полягає в тому, що в intra-блоках по-особливому обробляється старший коефіцієнт перетворення, що описує середній по блоку рівень сигналу (DC-коефіцієнт): кодується не сам коефіцієнт, а різниця між ним і DC-коефіцієнтом деякого сусіднього блоку. Схема кодування решти всіх коефіцієнтів виглядає таким чином. Здійснюється обхід блоку в зигзагоподібному порядку - від старшого коефіцієнта, що знаходиться в лівому верхньому кутку блоку, до молодшого, такого, що знаходиться в правому нижньому кутку (передбачено три варіанти зигзагоподібного обходу). В процесі обходу послідовно кодуються групи коефіцієнтів наступного типу: послідовність нульових коефіцієнтів і перший наступний за ними ненульовий коефіцієнт. Групи нумеруються, причому один номер резервується для специфічного випадку - ситуації, коли всі коефіцієнти є нульовими. Груповим номерам зіставляються коди змінної цілої довжини. Довжина вибирається виходячи з наступного критерію: номерам груп, що зустрічаються частіше, ставляться у відповідності коротші коди. Система кодів є префіксною: коди, що належать системі, не є початком інших кодів з цієї системи. Спосіб кодування блоків коефіцієнтів, отриманих в результаті застосування дискретного косинусного перетворення, обумовлений наступною особливістю: при обробці реальних зображень, як правило,

DCT-коефіцієнти убувають по абсолютній величині при русі по блоку зліва направо і зверху вниз; відповідно, під час зигзагоподібного обходу коефіцієнти убувають по абсолютній величині, причому, будучи квантованими, дуже багато з них (особливо коефіцієнти, що виявляються при обході останніми) мають нульове значення. Крім приведеного способу кодування, існує декілька модифікованих способів, при яких з коефіцієнтів верхнього рядка і лівого стовпця блоку заздалегідь віднімаються коефіцієнти, що стоять на тих же позиціях в сусідніх блоках. Стандартом передбачається дві схеми квантування коефіцієнтів, що отримуються в результаті дискретного косинусного перетворення. Одна з них повторює схему, яка використовується в стандарті H.263, і полягає в діленні всіх коефіцієнтів, окрім DC-коефіцієнта в intra-блоках, на одне і те ж число - подвоєний параметр квантування. При цьому у разі inter-блоків перед квантуванням з абсолютного значення коефіцієнта віднімається значення, рівне половині параметра квантування. DC-коефіцієнт в intra-блоці завжди ділиться на 8. У альтернативній схемі ретельніше враховуються особливості зорового сприйняття інформації. Для DC-коефіцієнта використовується нерівномірне квантування, при якому дільник залежить не тільки від параметра квантування, але і від типу колірної компоненти. Решта всіх коефіцієнтів кодується в два етапи. Спочатку кожен з них ділиться на відповідне йому число із стандартної для даного типу блоку (intra або inter) матриці квантування (розмір матриці співпадає з розміром блоку - 8x8). Матриця влаштована так, щоб при русі зліва направо і зверху вниз дільники збільшувалися. (Річ у тому, що зір сильніше сприймає зміни, що описуються низькими частотами гармонійного спектру, тому високими частотами часто можна безболісно нехтувати.) На другому етапі частково квантовані коефіцієнти повторно діляться на число, рівне подвоєному параметру квантування. Вектора руху кодуються покоординатно. Як і у випадку з DCT-коефіцієнтами, при генерації коду використовуються системи префіксних кодів змінної довжини. Кодується не сама координата вектора, а різниця між нею і одній з відповідних координат вектора деякого сусіднього блоку (у виборі беруть участь координати векторів трьох сусідніх блоків; вибирається середня з трьох координат). Різним значенням різниці координат зіставляються різні коди змінної довжини, при цьому маленьким по абсолютному значенню різницям відповідають коротші коди. Описані методи кодування коефіцієнтів і векторів руху задіються не завжди: виникають ситуації, коли дані процедури взагалі не потрібні. Однією з таких ситуацій є випадок рівності нулю всіх DCT-коефіцієнтів блоку. Про те, виконується це умова чи ні, сигналізує спеціальний прапорець. Крім цього прапорця, існує також і інший прапорець, наявність якого дозволяє спростити процедуру кодування. Він сигналізує про те, чи кодується даний inter-макроблок взагалі. Якщо прапорець має одиничне значення, ніяка додаткова інформація, що описує коефіцієнти і вектора руху не передається: в якості вектора руху беруться нульові вектори, також нульовою вважається помилка прогнозу. У стандарті обмовляються два методи пост-обробки відеопослідовності: деблокінг (deblocking) і дерингінг (deringing). Перший застосовується для зменшення блокового ефекту - особливого типу помилки відновлення закодованого зображення, обумовленого поблочним кодуванням з високим рівнем втрат; другий є технікою адаптивного згладжування сигналу, що дозволяє усувати окремі високочастотні спотворення. Серед іншого, стандарт

MPEG4 описує формат представлення деяких вельми специфічних візуальних об'єктів: нерухоме зображення (still image), лицьова анімація (face animation) і сітчастий об'єкт (mesh object). Нерухоме зображення (як правило, звичайна картинка) кодується із застосуванням вейвлет-перетворення. Здійснюється стандартна двомірна частотна декомпозиція, при цьому використовується біортогональний базис (9,3), або якийсь специфічний базис, параметри якого заздалегідь невідомі і передаються разом з кодом. Коефіцієнти, отримані в результаті вейвлет-перетворення, квантуються з використанням рівномірного скалярного квантування, а потім кодуються по стандартній схемі, в якій застосовуються так звані нуль-дерева (zero-tree). Код генерується на основі високоефективного арифметичного кодування. У стандарті описується достатньо оригінальна технологія, призначена для опису міміки людського обличчя. Обличчя відновлюється на основі його параметричної моделі (лицьова анімація). Уніфікований опис дозволяє представляти особу в достатньо економній формі в порівнянні із звичайними методами кодування описаними вище. Найбільш вірогідна область застосування такого рішення - кодування відеоконференцій. Спеціальне представлення об'єктів на площині у вигляді трикутної сітчастої структури з накладеною на неї текстурою носить назву mesh-технологія. Опис топології сітки здійснюється шляхом кодування координат вершин трикутників, а також векторів руху, що визначають зміну положення цих вершин в часі. Паралельно з трансформацією топології сітки проводиться трансформація текстури, що накладається. Сітчасте представлення об'єктів ефективно в тих випадках, коли ці об'єкти спочатку задаються у векторному вигляді.

3.4.1 Стандарт H.264

При розробці стандарту вперше були об'єднані зусилля фахівців Міжнародного Телекомунікаційного Союзу (International Telecommunication Union - ITU) в особі їх сектора (ITU-T) стандартизації, Міжнародній Організації по Стандартизації (International Organization for Standardization - ISO) і Міжнародній Комісії з Електротехніки (International Electrotechnical Commission - IEC). До об'єднання перша організація займалася розробкою телекомунікаційних стандартів, тоді як останні дві -стандартів загального плану. Раніше ними були запропоновані стандарти кодування відеоінформації H.261, H.263 (ITU-T); JPEG, JPEG2000, MPEG (ISO/IEC). На відміну від стандарту MPEG4, технології, що описуються в стандарті H.264, призначені для обробки звичайних відеопослідовностей, заздалегідь не розділених на відеоплощині. Природно, це не є обмеженням, оскільки всі методи кодування можуть бути з тим же успіхом застосовані і до випадку багатоплощинного кодування. Описувані в новому стандарті методи в цілому не сильно відрізняються від методів, передбачених стандартом MPEG4. Проте новий стандарт включає декілька достатньо перспективних рішень які заслуговують найпильнішої уваги. Як базові одиниці кодування в даному випадку виступають блоки розміру 4×4, 8×8, 8×4 і 4×8. При цьому розмір макроблоку залишається тим же - 16×16, що дає можливість здійснювати компенсацію руху з великою кількістю різних варіантів розбиття макроблоку на блоки. Розбиття здійснюється в два етапи. Спочатку макроблок 16×16 розбивається на одну або декількох прямокутних областей, кожна з яких має свій, відмінний від інших вектор

руху (можливі наступні варіанти розбиття: 16×16, $16 \times 8 + 16 \times 8$, $8 \times 16 + 8 \times 16$ і $8 \times 8 + 8 \times 8 + 8 \times 8 + 8 \times 8$). На другому етапі кожен підблок розміру 8×8 у свою чергу розбивається на області одним з перелічених способів: 8×8,8×4 + 8×4,4×8+ 4×8 і 4×4+ 4×4+ 4×4+ 4×4. Для блоків 4×4 застосовується вельми специфічне перетворення, відмінне від дискретного косинусного перетворення. Дане перетворення володіє ефективністю схожою з дискретним косинусним перетворенням, але з меншою обчислювальною складністю (використовуються тільки операції складання і зрушення). Вельми оригінальною ідеєю в даному випадку є повторне кодування старших коефіцієнтів перетворення. Кодування кожного блоку є повторне застосування 24 процедур кодування блоків (16 для компоненти Y і по 4 для компонентів U і V). Отримані 16 старших коефіцієнтів перетворення для компоненти Y і, відповідно, по 4 старших коефіцієнта для компонентів U і V піддаються повторному перетворенню. Для блоку старших коефіцієнтів компоненти Y використовується декілька відмінне від попереднього перетворення 4×4, а для блоку старших коефіцієнтів компонентів U і V спеціальне перетворення 2×2. Для блоків 8×8, 8×4 і 4×8 також передбачені свої перетворення, які, проте, не є обов'язковими. Замість кодування на основі перетворення може застосовуватися екстраполяція. При використанні такого рішення значення яскравості/кольору усередині блоку отримуються шляхом наближення, в якому враховуються значення яскравості/кольору граничних точок сусідніх блоків. Як видно, компенсація руху зазнала досить істотні зміни в порівнянні із стандартом MPEG4. Слід виділити також ще дві його особливості: велика точність представлення векторів руху і інший спосіб збільшення масштабу опорних фреймів, які використовуються для отримання прогнозу. Допускається точність,яка дорівнює або 1/4, або 1/8 (у MPEG4 максимальна точність, як відомо, складає 1/4). Для масштабування опорних фреймів застосовується достатньо складна багатоточкова інтерполяція. У сукупності дані рішення дозволяють отримати досить вагомий приріст в ефективності. Великий інтерес викликає метод кодування квантованих коефіцієнтів перетворення. Для intra-макроблоків/блоків застосовується так званий intra-прогноз. Із значень коефіцієнтів intra-макроблоків/блоків віднімаються числа, що отримуються з совокупностей значень деяких коефіцієнтів сусідніх макроблоків/блоків з використанням однієї з декількох схем прогнозу (для макроблоків передбачено 4 схеми, а для блоків - 9 схем). При кодуванні отриманих різниць для випадку intra-бокам і помилок прогнозу для випадку inter-блоків задіюється метод, практично ідентичний методу, який описано в стандарті MPEG4. Дієвим прийомом збільшення ефективності представлення відеоінформації виявляється використання для генерації коду арифметичного кодування замість префіксного. Особливістю пропонованої реалізації арифметичного кодування є те, що кодування приймає на вхід тільки бінарні символи. З одного боку, це декілька ускладнює застосування даного способу генерації коду (доводиться використовувати бінарну декомпозицію для багатосимвольних алфавітів), з іншого боку, подібний підхід робить можливими достатньо швидкі реалізації і полегшує процес побудови імовірнісних моделей. Стандарт H.264 спочатку був орієнтований на застосування в комунікаційних системах, тому особлива увага тут приділена підвищенню перешкодостійкості і забезпеченню високого ступеня гнучкості інформаційного уявлення, необхідної для забезпечення зручності переда-

чі закодованої відеоінформації по мережах різної природи. Передбачається особливі алгоритми розподілу закодованого потоку на блоки, що забезпечують необхідні характеристики інформаційного уявлення для кожного конкретного випадку.

Стандарт дійсно є помітним кроком вперед в порівнянні з поточною версією стандарту MPEG4. Формат кодування дозволяє добиватися ефективності представлення інформації, що перевершує ефективність, допустиму в рамках стандарту MPEG4, більш ніж на 30%. Природно, обчислювальна складність пропонованих методів декілька вище ніж у MPEG4.

Бібліоґрафія

[1] *Chui Ch.K.*An Introduction to Wavelets.- San Diego: Academic Press, 1992.
Чуи К. Введение в вейвлеты.- М.: Мир, 2001.

[2] *Daubechies I.* Ten lectures on wavelets.- SIAM, Philadelphia, 1992.
Добеши И. Десять лекций по вейвлетам.- Ижевск: НИЦ "Регулярная и хаотическая динамика", 2001.

[3] *Golomb S.* Run length encoding// IEE Transactions on Infirmation Theory, 12(7), 1966.- p. 399-401.

[4] *Gray R. and Neuhoff D.*Quantization // IEE Transactions on Infirmation Theory, 44(6), 1998.-p. 1-63.

[5] *Grossman A., Morlet J.* Decomposition of Hardy functions into square integrable wavelets of constant shape.- SIAM J.Math.Anal. 1984. 15.- C.723-736.

[6] *Haar A.* Zur theorie der orthogonalen functionensysteme.- Math.Annalen 1910. 69.- C.331-371.

[7] Image coding system JPEG 2000. Final committee draft version 1.0, 16 march 2000.- The ISO and ITU will provide cover pages.-2000.- 205 p.

[8] ISO/IEC JTC1/SC29/WG11, Coding of Moving Pictures and Audio: MPEG-4 Video Verification Model version 18.0, JTC1/SC29/WG11 N3908, Pisa, January 2001.

[9] Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, Joint Final Committee Draft (JFCD) of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)JVT- D157, Australia, July 2002.

[10] *Mallat S.A* Theory for Multiresolution Signal Decomposition: The Wavelet Representation".- IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 7, July 1989.- pp. 674-693.

[11] *Shapiro J.M.* Embedded image coding using zertrees of wavelet coefficients.- IEEE Trans.on Signal Processing. V.41 (1993), 12.- 3445-3462

[12] Subdivision for Modeling and Animation.- SIGGRAPH 2000 Course Notes.- 194 p.

[13] *Sweldens Wim.*The Lifting Scheme: A new philosophy in biorthogonal wavelet constructions. Wavelet Applications in Signal and Image Processing III.- Proc. SPIE 2569.- 1995.- pp. 68-79.

[14] *Turkowski Ken.* Filters for Common Resampling Tasks. Graphics Gems I.- Academic Press, 1990.- pp. 147-165.

[15] Wavelets and their Applications in Computer Graphics.- SIGGRAPH 95 Course Notes.- 239 p.

[16] *Warren Joe* Subdivision methods for geometric design.- Department of Computer Science Rice University.- 110 p.

[17] *Welch T. A.* A Technique for High-Performance Data Compression// Computer. - 1984. -Vol. 17. - N 6.

[18] *Ziv J., Lempel A.* An Universal Algorithm for Sequential Data Compression// IEEE Transactions on Information Theory. - 1977. - Vol. 23. - N 3.

[19] *Ziv J., Lempel A.* Compression of Individual Sequences via Variable Rate Coding// IEEE Transactions on Information Theory, - 1978. -Vol. 24. - N 5.

[20] *Ватолин Д.С.* Алгоритмы сжатия изображений.- Изд. МГУ, 1999.- 76 с.

[21] *Воробьев В.И., Грибунин В.Г.* Теория и практика вейвлет-преобразования.- СПб: ВУС, 1999.- 204 с.

[22] *Гонсалес Р., Вудс Р.* Цифровая обработка изображений .- М.: Техносфера, 2005.- 1070 с.

[23] *Павлидис Т.* Алгоритмы машинной графики и обработки изображений.- Москва: Радио и связь, 1986.- 400 с.

[24] *Петухов А.П.* Введение в теорию базисов всплесков.-СПб, Изд. СПбГТУ, 1999.

[25] *Порев В.М.* Комп'ютерна графика.-К.: "Корнійчук",2000.-256 с.

[26] *Фень Юань.* Программирование графики для Windows.- Изд. Питер, СПб, 2002.- 1070 с.

[27] *Лигун А.А., Шумейко А.А.* Асимптотические методы восстановления кривых.- Киев: Изд. Института математики НАН Украины, 1997.- 358 с.

[28] *Ligun A.A., Shumeiko A.A.* Linear method of recovery of function of two variables on a binary lamination. // East Journal of Approximation, (2001), v.7, N 3, 1-18.

[29] *De Marchi S., Ligun A., Timchenko S., Shumeiko A.* An interpolant defined by subdivision and the analysis of the error. // Journal of Comp. and Appl. Mathematics, USA, 145 (2002), p. 71 – 88.

[30] *Ligun A.A., Shumeiko A.A., Radzevitch S.P., Goodman E.D.* Asymptotically Optimal Disposition of Tangent Points for Approximation of Smooth Convex Surfaces by Polygonal Functions. // Computer Aided Geometric Design, USA, 14, 1997, p. 1-14.

[31] *Ligun A.A., Shumeiko A.A., Radzevitch S.P., Goodman E.D.* Asymptotically Optimum Recovery of Smooth Contours by Bézier Curve. // Computer Aided Geometric Design, USA, 1998, 15, p. 495 – 506.

[32] *Лигун А.А., Шумейко А.А., Журба В.Н.*Об оптимальном методе бинарного увеличения изображения // Математичне моделювання, 1,2 (15), 2006.- с. 21 - 25.

[33] *Лигун А.А., Шумейко А.А., Коротков В.С.* О контроле качества сложных изображений // Юбилейный сборник научно-технических трудов ДГТУ.- 1995.- С.312-318.

[34] *Шумейко О.О., Лигун О.О., Батрак Д.С.* Поліпшення якості фотозйомки і відеозапису при проведенні оперативно-розшукових заходів // Науковий вісник Дніпропетровського юридичного інституту МВС України .- Дніпропетровськ, 2000 .- N3.- С.298-302.