КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.627

В.І. БОВСУНІВСЬКИЙ

ТЕХНОЛОГІЯ ОПЕРАТИВНОЇ ФІЛЬТРАЦІЇ І СТИСКУ ВІДЕОДАНИХ У РЕАЛЬНОМУ ЧАСІ

Вступ. Ефективна обробка відеоданих у реальному часі є важливою умовою, яка необхідна для можливості передачі відеозапису з високою роздільною здатністю та високої якості. Більшість відеосенсорів не підтримують формат зйомок у такому режимі. Основною проблемою цього є використання сучасних алгоритмів стиску, які базуються на ортогональних перетвореннях та потребують багато часу для конвертування даних. При отриманні даних у реальному часі (відеозйомка) відеосенсори накопичують інформацію у буфери, після цього здійснюють обробку. Проте при певній роздільній здатності буфер буде постійно збільшуватись, що призведе до повного вичерпання вільного місця і тоді відеосенсор буде не спроможний обробляти дані.

Мета роботи – розробка комплексу алгоритмів та побудова технології обробки відеоданих, яка дасть можливість обробляти відеодані високої роздільної здатності, використовуючи мінімум часу на обробку та досягши достатнього ступеня стиску.

Як умови отримання даних будемо мати дискретний RGB сигнал. В кожен момент часу t будемо мати три відліки складових R, G і B. Кожен відлік незалежно буде фільтруватися та піддаватися стиску, після чого записуватися на носій.

Розпаралелювання обчислень. Оскільки кожен з трьох отриманих компонентів сигналу від відеорегістратора обробляється однаково, для кожної компоненти будемо застосовувати один і той же алгоритм фільтрації та стиску даних.

V. Bovsunivsky

THE TECHNOLOGY OF RAPID FILTRATION AND COMPRESSION OF VIDEO DATA IN REAL-TIME

The author describes the technology of rapid filtration and compression of video data in real time using parallel calculations.
Key words: video compression, video encoding.

Описана технология оперативной фильтрации и сжатия видеоданных в реальном времени с использованием разпараллеливания вычислений.
Ключевые слова: сжатие видео, кодирование видео.

Описана технологія оперативної фільтрації і стиску відеоданих у реальному часі з використанням розпаралелювання обчислень.
Ключові слова: стиск відео, кодування відео.

Враховуючи сучасні тенденції розвитку розпаралелювання обчислень, вже наявність 4-х ядерного процесора дозволяє нам ефективніше обробляти сигнали кожної з трьох компонентів в окремому потоці. Це в тричі прискорить процес обробки відео на відміну від використання одноядерного процесора. В такому випадку весь процес кодування та декодування будемо розглядати у випадку однієї компоненти.

Конвертація даних. Як оброблювані дані можна використати переконвертовані дані про відліки в YUV форматі. YUV – формат представлення кольору з використанням трьох компонент: яскравості та двох різницевих компонент. Завдяки особливостям сприйняття кольору людського ока (людське око більш чутливе до яскравості ніж до зміни кольору), при стиску і фільтрації різницевих компонент можна допускати більш значну похибку, ніж при кодуванні зображення в RGB-форматі. Для конвертації пікселів між форматами RGB та YUV використовують лінійні формули:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (1)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}. \quad (2)$$

За рахунок лінійності, швидкість перекодування досить висока, проте вагомим недоліком цієї схеми є втрата даних за рахунок округлення при конвертації в обох напрямках. Наприклад, хроматичний червоний колір (U) отримуємо за формулою:

$$U = -0.14713 * R - 0.28886 * G + 0.436 * B. \quad (3)$$

Підставимо довільну точку RGB (130; 86; 213), отримуємо $U = 48{,}89914$. Проте нам доведеться відкинути дробову частину, оскільки нам потрібно оперувати цілими числами. Аналогічно отримуємо хроматичний синій колір та яскравість, маємо: YUV (114; 49; 14). При перетворенні точки назад за допомогою формул (2) отримуємо точку RGB’ (130; 86; 214). Як бачимо перетворення RGB в RGB’ досить точне, проте існує невелика похибка, приблизно рівна 1%, яка візуально не впливає на якість зображення [1].

Отримання YUV сигналу може відбуватися двома шляхами – апаратно та програмно. У першому випадку відео регістратор буде видавати готовий переконвертований в YUV сигнал, в іншому випадку – це потрібно буде робити програмно за формулами (1) та (2).

Оскільки ми прийняли, що обробка кожного з відліків відбувається паралельно, потрібно, щоб кожен паралельний процес, який закінчив обробку попередніх даних мав доступ до наступних даних, причому ці дані вже мають бути пе-

реконвертовані в YUV формат (рис. 1). Тому має існувати ще один потік, який буде готувати дані для подальшої конвертації [1].

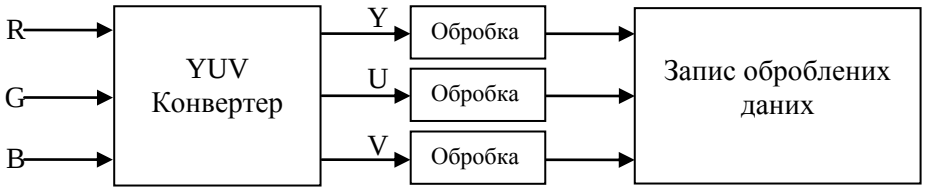

РИС. 1. Схема паралельної обробки даних

Фільтрація. Зазвичай кадри відео, як і зображення, містять надлишкову інформацію, яка не сприймається оком людини. Такі дані можна відфільтрувати, використовуючи різні фільтри для отримання кращого результату стиску в подальшому. В режимі оперативного отримання даних, необхідно використовувати фільтр, який дозволятиме при мінімальних затратах часу отримувати хороший результат фільтрації даних. Визначення суттєвих відліків слід здійснювати динамічно, в залежності від поведінки кривої. Однак слід зауважити, що методи визначення коефіцієнтів для визначення суттєвих відліків відрізнятимуться для кривої яскравості та хроматичних кольорів [1].

Визначення суттєвих відліків буде відбуватися наступним чином: нехай маємо вісь T – часова вісь, на якій у кожен момент часу t_i маємо $F(t_i)$ – значення відліку в момент часу t_i. Вважатимемо початковий відлік $F(t_0)$ суттєвим. Нехай маємо суттєвий відлік $F(t_n)$. Тоді наступним суттєвим відліком буде $F(t_{n+m})$, для якого виконується умова:

$$|F(t_n) - F(t_{n+m})| > H_0, \text{і} \; \forall s, s \in N, n < s < m, |F(t_n) - F(t_{n+s})| \leq H_0, \quad (4)$$

де H_0 – максимальна амплітудна відстань між відліками, яка вважається не суттєвою.

Використовуючи формулу (4), ми зможемо знайти всі суттєві відліки, які відповідають H_0 (рис. 2).

Очевидно, чим більший H_0, тим менше суттєвих відліків відберемо і тим гірша якість зображення буде у нас, і навпаки, чим менший H_0, тим більше отримуємо суттєвих відліків і тим точніше будуватимуться криві. Саме така лінійна модель фільтрації дає високу швидкість обробки даних і хороший результат фільтрації кривої [2].

Підібравши оптимальний H_0 та застосовуючи фільтрацію до кривої, отримаємо відфільтровану криву і суттєві відліки на цій кривій. Проте існують випадки, коли H_0 не забезпечуватиме достатню фільтрацію та пропускатиме несуттєві відліки. Також можуть бути випадки, де фільтруватимуться важливі нам відліки,

для таких випадків слід зменшити значення H_0. Для цього H_0 слід визначати адаптивно, в залежності від різних факторів, серед яких:

- загальний фон огинаючих на ділянці;
- рівень зашумленості кадру;
- тип огинаючої;
- необхідна якість відео.

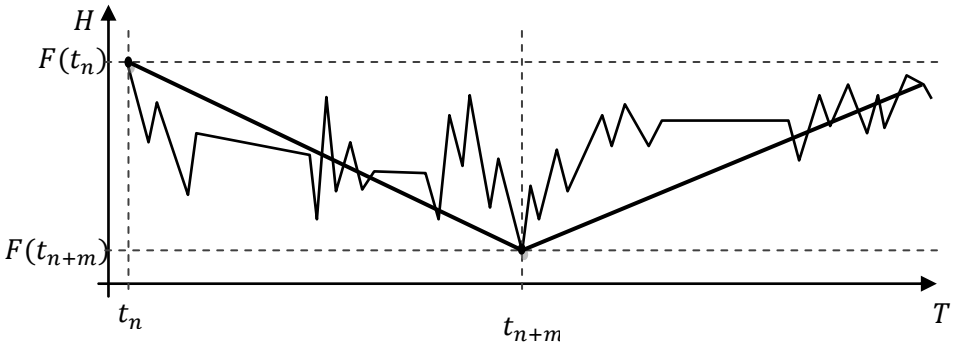

РИС. 2. Визначення суттєвих відліків

Загальний фон огинаючої на певній ділянці – це поведінка кривої, яка зазвичай описує один і той самий об'єкт на відео. Якщо об'єкт має якийсь один колір чи має строкату форму, то і крива відліку буде однакова на всій ділянці кадру, де знаходиться цей об'єкт. У зв'язку з розмаїттям можливих об'єктів, які можуть потрапити на кадр, їх положення, освітлення на кадрі, прокласифікувати їх усіх неможливо [3]. Проте вдалося виділити групу об'єктів, яка дуже часто зустрічається і стиск таких об'єктів дає хороший результат. Це великі об'єкти з монотонним забарвленням, з плавним градієнтним переходом кольору (монотонний фон, безхмарне небо, хмарне небо, водяна гладь та ін.) [4]. Ці та інші об'єкти можна виділити за однаковою поведінкою кривої на всій протяжності цього об'єкта. Для цього в процесі фільтрації слід визначати максимальну та мінімальну амплітуду кривої $h_{\max}$ та $h_{\min}$ відповідно.

$$h_{\max} = \max_{n < i < n+\delta} |F(t_n) - F(t_i)|,$$
$$\forall\, t_i \in (t_n, t_{n+\delta}), |F(t_n) - F(t_i)| \geq h_{\min_0}, \qquad (5)$$

де t_n – суттєвий відлік; $\delta = \min(q, m)$; q – поріг зміни значення H_0 на $h_{\max}$; t_{n+m} – наступний суттєвий відлік; $h_{\min_0}$ – мінімально допустиме значення H_0.

$$h_{\min} = \min_{n < i < n+p} |F(t_i) - F(t_{i+1})|,$$
$$\forall\, t_i \in (t_n, t_{n+p}),\ H_0 < |F(t_i) - F(t_{i+1})| \leq h_{\max_0}, \qquad (6)$$

де t_n – суттєвий відлік; p – поріг зміни значення H_0 на $h_{\min}$; t_{n+m} – наступний суттєвий відлік; $h_{\max_0}$ – максимально допустиме значення H_0.

Значення H_0, $h_{\min_0}$, $h_{\max_0}$, p та q задаються на початку обробки даних, як початкові параметри. Очевидно, що H_0 може адаптивно змінюватися в залежності від поведінки кривої, проте воно завжди $h_{\min_0} \leq H_0 \leq h_{\max_0}$. Максимальне значення порогу фільтрації $h_{\max_0}$ має бути встановлено таким чином, щоб не допустити фільтрацію корисних даних. Якщо всі амплітудні значення кривої, більші за H_0 на протязі часу p, то H_0 буде збільшено до мінімального значення амплітуди, знайденого на проміжку (t_n, t_{n+p}). Проте, якщо на проміжку (t_n, t_{n+p}) трапиться $|F(t_i) - F(t_{i+1})| < H_0$, пошук мінімальної амплітуди буде перерваний, та буде розпочатий спочатку, коли знову почне виконуватися умова (6). І лише, якщо умова (6) буде виконуватися для всіх t_i на проміжку (t_n, t_{n+p}), H_0 буде збільшено (рис. 3).

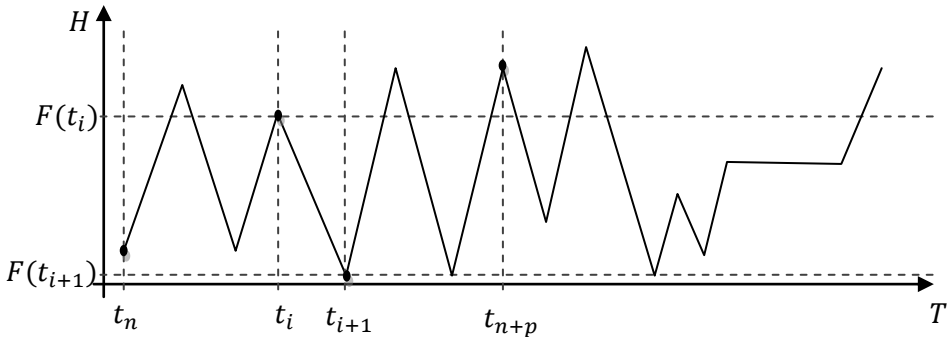

РИС. 3. Адаптивне збільшення відстані H_0

Таким чином поріг фільтрації буде збільшений на допустиму величину і здійснюватиме фільтрацію даних на динамічних проміжках кривої, які свідчать про строкату форму об'єкта на відео або про наявність шумів. В обох випадках часткова фільтрація не вплине на загальну картину такого забарвлення. В іншому випадку, різке короткочасне збільшення амплітуд не буде відфільтроване, оскільки адаптивний фільтр не встигне налаштуватися на потрібну амплітуду, тому такі об'єкти, як грані, краї та різки переходи між об'єктами практично не будуть піддаватися фільтру [4].

Якщо динаміка кривої значно зменшилася, на протязі часу q, H_0 буде зменшено до максимального значення $|F(t_n) - F(t_i)|$, якщо в кожній точці проміжку (t_n, t_{n+q}), $|F(t_n) - F(t_i)| < H_0$. В іншому випадку, при $|F(t_n) - F(t_i)| \geq \geq H_0$, отримаємо суттєвий відлік і перевірка виконання умови розпочнеться з поточного моменту. В будь-якому випадку H_0 не може бути менше $h_{\min_0}$ (мінімальний поріг). Значення $h_{\min_0}$ має обиратися таким чином, щоб будь-яку криву,

амплітудні значення якої не перевищують $h_{\min_0}$ – можна було замінити прямою, без помітної (суттєвої) втрати даних (рис. 4). Таким чином $h_{\min_0}$ встановлює мінімальний поріг фільтрації, нижче якого дані будуть відфільтровані в будь-якому випадку. Амплітудні значення кривої, більші за $h_{\min_0}$ будуть відфільтровані, в залежності від H_0 , яке в свою чергу залежить від поведінки кривої.

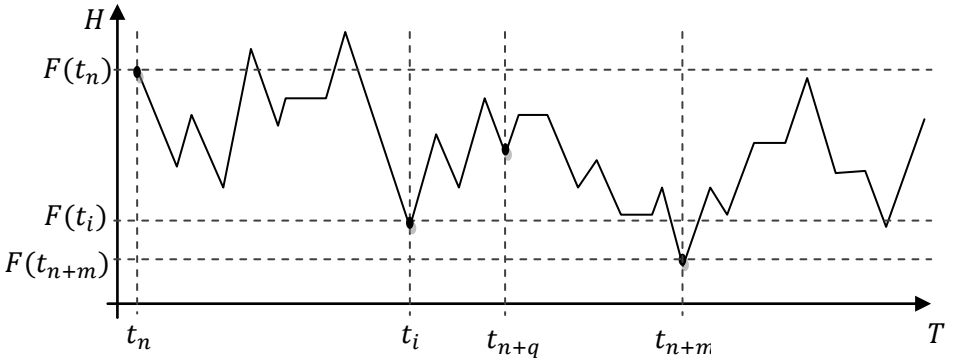

РИС. 4. Адаптивне зменшення відстані H_0

Стиск. Після адаптивної фільтрації здійснюється відбір суттєвих відліків, які є точками зміни напрямку кривої, тобто є центрами зміни поведінки кривої. Визначення звичайних точок екстремуму (мінімум, максимум та точки перегину) є недостатнім для певних кривих, оскільки існують такі криві, при відновленні яких буде спостерігатися значна різниця між відновленими відліками. Наприклад, крива, показана на рис. 5 буде мати локальний максимум в будь-якій точці $\boldsymbol{t_i} \in [\boldsymbol{t_{j-l}}, \boldsymbol{t_{j+k}}]$, тобто будь-яка точка з цього сегменту може бути взята, як ключова точка для відновлення даних, проте відновлена крива буде суттєво відрізнятися від початкової кривої. Тому потрібно враховувати не просто точки екстремуму, а всі точки зміни напрямку кривої.

Нехай маємо точку на відфільтрованій кривій у момент часу t_i. Точка $F(t_i)$ називається точкою зміни напрямку кривої, якщо виконується умова:

$$F(t_{i-1}) - F(t_i) \neq F(t_i) - F(t_{i+1}). \quad (7)$$

Нехай маємо точку на відфільтрованій кривій у момент часу t_i. Точка $F(t_i)$ називається точкою зміни напрямку кривої з мінімальним відхиленням θ, якщо виконується умова:

$$|2F(t_i) - F(t_{i-1}) - F(t_{i+1})| > \theta. \quad (8)$$

Відбирати і зберігати будемо лише ті суттєві відліки, для яких виконується умова (8). Значення θ будемо встановлювати як параметр перед запуском обробки даних. Умова (8) перетворюється в умову (7) при $\theta = 0$. Оскільки $F(t_i) \in N$, то з формули (8) видно, що і $\theta \in N$, де N – множина натуральних чисел. Також з нашої формули можна встановити, що θ не може бути більше 510, проте досліди показали, що не варто встановлювати для θ значення більше 50, оскільки тоді втрачаються важливі дані, відео стає помітно спотвореним.

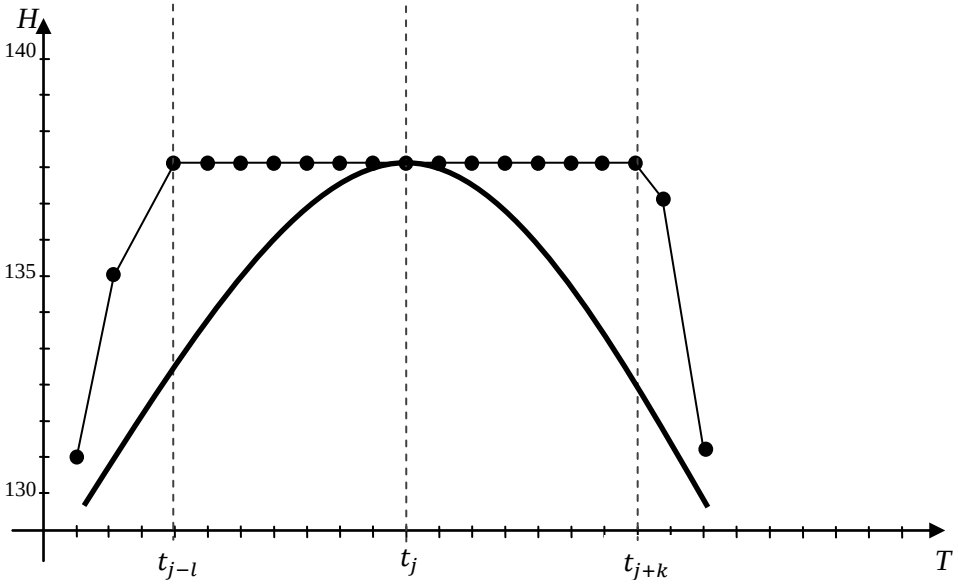

РИС. 5. Похибка при відновленні кривої за точками екстремуму оригінальної кривої (жирною лінією позначено відновлену криву, звичайною – оригінальну)

Формат збереження даних. Оскільки обробка кожної компоненти відбувається паралельно, то для забезпечення оперативності обробки даних, для кожної компоненти буде використовуватися 3 окремі файли. При збереженні даних кожної компоненти нам потрібно зберегти інформацію про дані, такі як розміри кадрів, джерело, дати, інформація про синхронізацію та ін. Інформація про синхронізацію забезпечуватиме зв'язок між усіма трьома компонентами та не допустить підміни якоїсь з компонент [5]. Для такої інформації достатньо буде 180 байтів. Далі у файл записуватимуться пари VH відліків (V) та відстані до наступних відліків (H). На кожну таку пару знадобиться по два байти. Очевидно, що максимальна відстань між відліками не може бути більшою за 255, тому при стиску, якщо на протязі 255 одиниць часу не трапилося жодного суттєвого відліку, потрібно робити контрольний відлік.

Зустрічатимуться такі випадки, коли велика кількість сусідніх H рівна 0. А оскільки H займає цілий байт, такі дані не стиснуться, а їх розмір збільшиться в двічі. Тому у випадку, якщо трапиться як мінімум дві пари VH (рис. 6.) в яких $H = 0$, вони будуть замінені на сегмент $FVVF$, де F – зарезервований байт початку та закінчення сегменту, в якому всі відстані рівні 0. Як зарезервований байт можна взяти 255, а всі суттєві відліки при фільтрації заміняти на 254. Така заміна не призведе до візуального зниження якості зображення.

VH**VHVH** ... **VH**VHVH → VH**FVV** ... **VF**VHVH

РИС. 6. Заміна формату зберігаємих даних у випадку коли відстані H рівні 0 (жирним виділені відліки з $H = 0$)

Відновлення. При вирішенні задачі відновлення даних, маючи масиви даних, в яких міститься інформація про точки, виникає необхідність побудови деякої функції $y(x)$, яка без проблем могла б відновити несуттєві відліки максимально наближені до початкового їх положення. Для отримання $y(x)$необхідно побудувати інтерполяційну залежність у проміжках між суттєвими відліками, а також забезпечити крайові умови. За визначенням інтерполяція означає побудову функції $f\ (x)$, що апроксимує залежність $y(x)$ у проміжних точках. У більшості практичних застосувань, як і в нашому випадку, суттєві відліки слід з'єднати не ламаною лінією (як у випадку лінійної інтерполяції), а гладкою кривою. Тому краще всього для таких цілей підходить інтерполяція сплайнами. Сенс сплайн-інтерполяції полягає у тому, що на кожному кроці інтерполяції здійснюється апроксимація у вигляді певної поліноміальної залежності $f\ (x)$ [6]. При цьому для кожного кроку виходить свій поліном, і його коефіцієнти підбираються такими, щоб на кордонах кроку виконувалися крайові умови. А саме, якщо застосовуються сплайни у вигляді поліномів ступеня m, то нескладно показати, що їх коефіцієнти можна вибрати так, щоб забезпечити безперервність похідних порядку до $(m - 1)$-ї включно. Оскільки операція відновлення не вимагає суттєвої оперативності, і час відновлення даних залежить уже не від швидкості потоку даних, а від потужності параметрів апаратури [5].

Досліди показали, що найкращі результати при відновленні дають саме кубічні сплайни, тобто поліноми третього ступеня:

$$f(x) = ax^3 + bx^2 + cx + d. \qquad (9)$$

Коефіцієнти a, b, c, d розраховуються незалежно для кожного проміжку інтерполяції. Для розрахунку кубічної сплайн-інтерполяції (9) на проміжку між x_i та x_{i+1} обчислюється таким чином:

$$f(x) = \frac{(x_{i+1} - x)^2(2(x_i - x) + h)y_i}{h^3} + \frac{(x_{i+1} - x)^2(x - x_i)m_i}{h^2} +$$
$$+ \frac{(x_{i+1}-x)^2\,(2(x_i-x)+h)y_{i+1}}{h^3} + \frac{(x_{i+1}-x)(x-x_i)^2 m_{i+1}}{h^2}, \quad (10)$$

де

$$m_i = \frac{(y_{i+1}-y_{i-1})}{2h}, \quad (11)$$

$$m_0 = \frac{(-3y_0+4y_1-y_2)}{2h}, \quad (12)$$

$$m_N = \frac{(3y_N-4y_{N-1}-y_{N-2})}{2h}. \quad (13)$$

Кубічна сплайн-інтерполяція забезпечує рівність у вузлах не тільки самих сусідніх параболічних сплайнів, але і їх 1-х та 2-х похідних. Завдяки цьому сплайн-інтерполяція виглядає дуже гладкою (рис. 7). Формули (9 – 13) – це не єдиний спосіб побудови кубічних сплайнів. Є певна свобода у визначенні сплайн-інтерполяції, яка в переважній більшості завдань не призводить до значимих змін кривої [6].

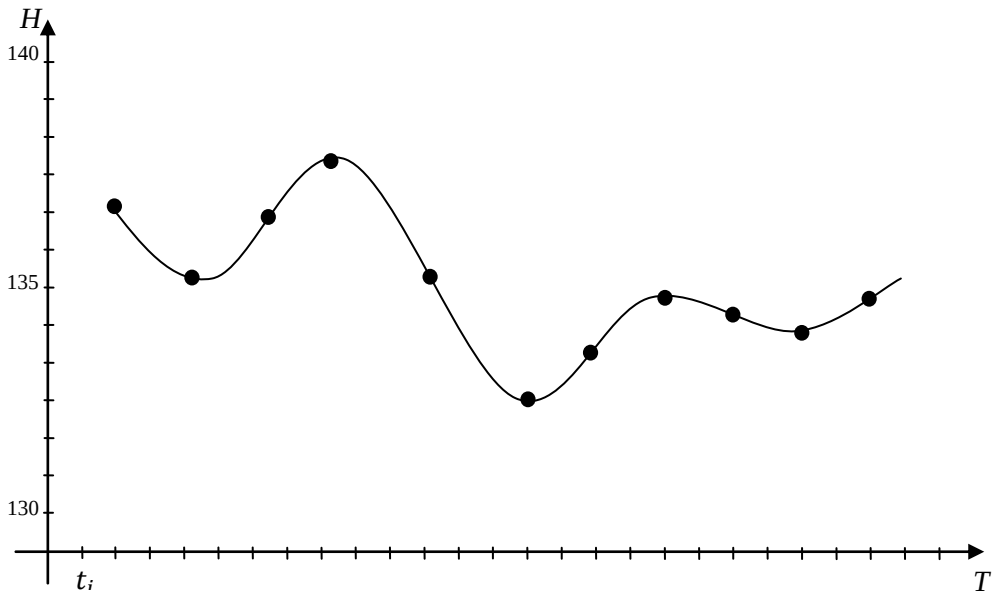

РИС. 7. Відновлення кривої за суттєвими відліками

Висновки. Даний метод обробки даних в оперативному режимі орієнтований у першу чергу на швидкість обробки даних при помірному стиску. Завдяки такому підходу можна стискати широкоформатні відеодані та інші громіздкі відеосигнали в процесі їх отримання. Висока швидкість обробки даних досягається за рахунок використання лінійних методів фільтрації та розпаралелювання процесів обробки кожного відліку окремо. Технологія використовує апаратні можливості чотирьох-ядерного процесора, який сприяє прискоренню обробки даних. Завдяки особливим підходам алгоритму до обробки різних видів кривих [4] та спеціального методу стиску, використання алгоритму дає непоганий результат стиску даних.

1. *Poynton C.* Digital Video and HDTV Algorithms and Interfaces. – San Francisco: Morgan Kaufmann, 2003. – 736 p.
2. *Бовсунівський В.І.* Нелінійний стиск та відновлення відліків обвідної сигналів та відеосигналів // Матеріали проблемно-наукової міжгалузевої конф. "Інформаційні проблеми комп'ютерних систем, юриспруденції, енергетики, економіки, моделювання та управління (ПНМК-2010)". – Бучач: Бучачський інститут менеджменту і аудиту, 2009. – С. 17 – 22.
3. *Селомон Д.* Сжатие данных, изображений и звука. – М.: Техносфера, 2004. – 368 с.
4. *Бовсунівський В.І.* Дослідження спотворень огинаючих фрагментів відеосигналів // Штучний інтелект. – К.: Україніка наукова, 2011. – С. 41 – 45.
5. *Шевчук Б.М., Задірака В.К., Гнатів Л.О., Фраєр С.В.* Технологія багатофункціональної обробки і передачі інформації в моніторингових мережах. – К.: Наук. думка, 2010. – 370 с.
6. *Шелевицький І.В.* Методи та засоби сплайн-технології обробки сигналів складної форми. – Кривий Ріг: Європейський університет, 2002. – 304 с.

Отримано 20.10.2011