№ 1(3), 2021

УДК 004.93

Ю. Є. Кинаш[1], М. М. Коломоєць [1], В. М. Мищишин[2]
Національний університет "Львівська політехніка",
[1] кафедра інформаційних технологій видавничої справи,
[2] кафедра комп'ютеризованих систем автоматики

ОБРОБКА ГРАФІЧНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ АЛГОРИТМУ ФЛОЙДА-СТЕЙНБЕРГА

https://doi.org/

Розроблено систему для обробки графічних зображень із застосуванням алгоритму Флойда-Стейнберга. Розроблена система сприяє згладжуванню зображень при від обра-женні на пристроях із різною роздільною здатність та різним набором палітри кольорів. Внаслідок застосування алгоритму Флойда-Стейнберга оброблені зображення мають мінімальні спотворення при їхньому відтворені. Запропонований алгоритм можна застосувати для стиснення та передавання зображень і звукових сигналів. Система розроблена з використанням мови Java Processing.

Ключові слова – комп'ютерна графіка, алгоритм Флойда-Стейнберга, обробка зображень.

The system for processing graphic images using the Floyd-Steinberg algorithm has been developed. The developed system helps to smooth images when displayed on devices with different resolutions and different sets of color palettes. Due to the application of the Floyd-Steinberg algorithm, the processed images have minimal distortion during their reproduction. The proposed algorithm can be used to compress and transmit images and audio signals. The system is designed using the Java Processing language.

Keywords – computer graphics, Floyd-Steinberg algorithm, image processing.

1. Постановка проблеми

У 21 столітті людство здебільшого сприймає інформацію візуально через GIF анімації, відео-історії у соцмережах, скетч-відео, відео на YouTube та ін., причому, щоб доступ до зображень був в будь-який момент, незалежно від швидкості інтернету. Для зображення сьогодні притаманні висока роздільна здатність та значний об'єм, але не завжди у користувача наявний дисплей з потрібною роздільною здатністю, або з такою широкою гамою кольорів. Саме тому має сенс підлаштовувати зображення під певну палітру або зменшити його деталізацію чи роздільну здатність, щоб воно зберігало свої основні деталі без втрати якості або з незначною втратою якості. З цією метою застосовують алгоритми Erorr Difusion, які зменшують палітру кольору фотографії, зберігають різницю між старим та новим пікселем та розподіляють різницю між сусідніми пікселями..

Отже, методи обробки зображення є актуальною задачею для застосування на практиці.

2. Застосування алгоритмів перетворення зображень

Є ряд алгоритмів, які дозволяють стиснути, зменшити об'єм зображення, зменшивши об'єм кольорової палітри зображення. Також існує підвид таких алгоритмів, які грунтуються на відносній похибці, або також Error Difusion алгоритми. Вони застосовуються іноді в генераторах GIF-анімацій для того, щоб зменшити палітру зображення до базових 256 кольорів [1-3].

Псевдотонування залишається унікальним методом не тільки з практичної точки зору, наприклад, для підготовки повноколірного зображення при друці на чорно-білому принтері, а й також для художніх цілей. Дизеринг також знаходить застосування у веб-дизайні, де цей метод використовується для скорочення числа кольорів зображення, що зменшує розмір файлу і, відповідно, трафік без шкоди для якості самого зображення. Також цей підхід використовується при зменшенні цифрових фотографій у форматі RAW в 48 або 64 біти на піксель до RGB в 24 біти на піксель при редагуванні.

Проблеми виникають кожного разу, коли зображення відображається на пристрої, що підтримує менше кольорів, ніж містить зображення. Тонкі градієнти в оригінальному зображені можуть бути замінені плямами однорідного кольору, і в залежності від обмежень пристрою, вихідне зображення може суттєво змінитися. Псевдотонування або дизеринг є спробою вирішення цієї проблеми. Псевдотонування працює через наближене вираження недоступних кольорів доступними, для чого доступні кольори змішуються так, щоб імітувати недоступні. Порівняння різних підходів обробки зображень показано на рис.1.

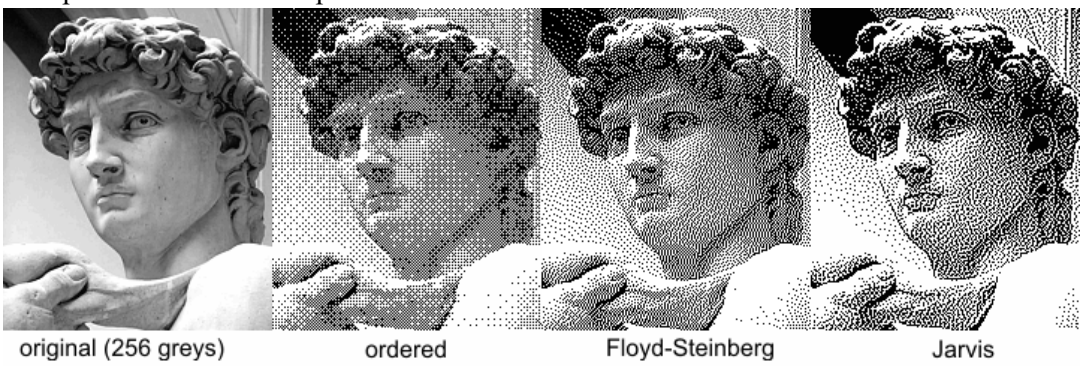

Рис. 1. Порівняння error difusion алгоритмів

Найчастіше на практиці застосовується алгоритм Флойда-Стейнберга. Реалізація даного алгоритму актуальна і зараз, бо інформація здебільшого сприймається візуально, а тому швидке завантаження зображень або анімацій відіграє ключову роль.

3. Мета роботи

Метою роботи є розроблення системи обробки графічних зображень із застосування алгоритму Флойда-Стейнберга.

4. Опис роботи системи

4.1. Вхідна та вихідна інформація

Вхідною інформацією при використанні системи виступає зображення до процедури обробки, а вихідною – результуюче зображення після застосування алгоритму дизерінгу.

4.2. Опис алгоритму задачі

Застосування алгоритму передбачає використання помилки при квантизації (Quantize Error Difusion), тобто при зменшенні палітри зображення, є певні показники, на які колір міг змінитися, внаслідок чого в певному співвідношенні цей показник Quantize Error застосовується на сусідні пікселі. Так як проходження відбувається з лівого верхнього кута зображення, тобтоз самого його початку, то зміни застосовуються на пікселі, що розташовані нижче та правіше від пікселя, котрий на даний момент обробляється.

Значення R, G, B можуть приймати значення від 0 до 255. Загалом це більше 16 мільйонів можливих комбінацій, але їх можна зменшити. Наприклад, найнижчою межею цього є 2 комбінації – або 0 або 255, тоді є лише 2*2*2=8 комбінацій кольорів. Існує фактор квантизації, який виражає на

скільки проміжків можна поділити комбінації кольорів. В результаті виходить фактор+1 комбінації вихідної палітри [1-3].

$$newPixel = round\left(\frac{factor*oldPixel}{255}\right) * floor(\frac{255}{factor}) \quad (1)$$

Випадок для квантизації кольору на прикладі GRAYSCALE-палітри з факторами 1 та 3 показаний на рис. 2.

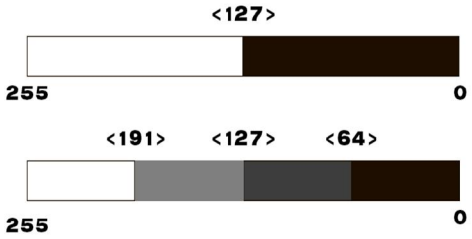

Рис. 2. Приклад квантизації кольору на прикладі GRAYSCALE-палітри з факторами 1 та 3

Найголовнішим у застосуванні алгоритму Флойда-Стайнберга є передача значень похибки квантизації сусіднім пікселям. Значення похибки квантизації визначається, як різниця між старими значеннями кольору та значеннями після квантизації:

До	255	150	10
Після	255	255	0
Q.error	0	-105	10

Рис. 3. Приклад визначення похибки квантизації

Це відбувається згідно формули:

$$quantError = \begin{matrix} oldRed - newRed \\ oldGreen - newGreen \\ oldBlue - newBlue \end{matrix} \quad (2)$$

Наступним кроком є розподіл результуючих значень між сусідніми пікселіми в наступних пропорціях:

$$\begin{bmatrix} & * & \frac{7}{16} & \dots \\ \dots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \dots \end{bmatrix} \quad (3)$$

де (*) – піксель, який в поточний момент обробляється.

Формули для розрахунків наведено нище:

$$\begin{matrix} pixel[x+1][y] = pixel[x+1][y] + quantError * \frac{7}{16} \\ pixel[x-1][y+1] = pixel[x-1][y+1] + quantError * \frac{3}{16} \\ pixel[x][y+1] = pixel[x][y+1] + quantError * \frac{5}{16} \\ pixel[x+1][y+1] = pixel[x+1][y] + quantError * \frac{1}{16} \end{matrix} \quad (4)$$

Відповідно до запропонованого алгоритму написано код програми для роботи із зображеннями.

4.3. Опис програмного продукту

Для створення програмного продукту застосовано середовище розробки з використанням мови програмування Java Processing, котра містить інструментарій для роботи з графікою та зображеннями [4-8]. Середовище Processing містить функції – setup і draw.

За допомогою функції Setup виконують підготовку даних до подальшої обробки. Застосування цієї функції забезпечує завантаження зображень.

Завданням функції Draw, котра виконується з частотою 30 разів на секунду, є відображення та оновлення зображення у режимі реального часу.

У програмному коді описаний клас QuantError, котрий відповідає за значення похибки у трьох колірних компонентах RGB.

Інтерфейс користувача обмежений лише відображенням двох зображень у вікні – зліва початкове, справа – після результату обробки. Розмір вікна задається подвійною шириною початкового зображення, висота рівна висоті зображення.

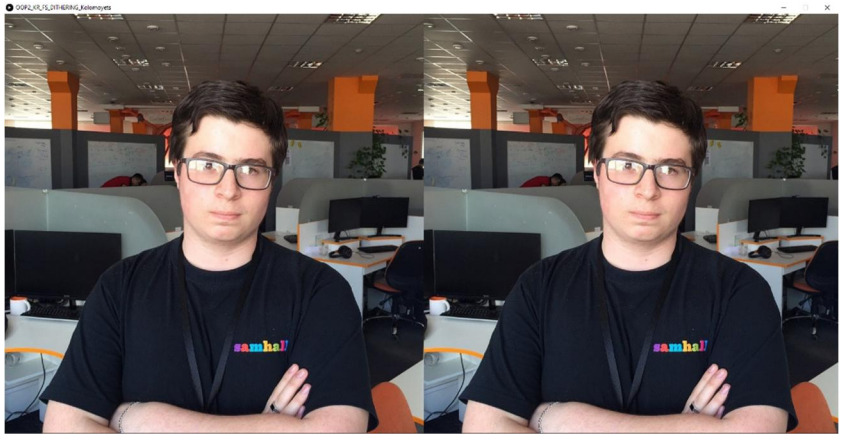

Рис. 4. Приклад відображення вікна програми

Програма має папку data, котра є обов'язковою для проекту, якщо потрібно виконати завантаження певних даних у програму з стандартного шляху. У разі знаходження даних в іншій папці буде видано помилку виконання програми. При використанні Processing слід мати на увазі для якої з версій буде використано програму: x32(x86) та x64. У кожному проекті міститься файл .exe, папки з бібліотеками та іншими файлами, які необхідні для правильного запуску та папка data з стартовими даними. Для використання програми слід вставити своє зображення в папку data і після запуску програми почекати завершення роботи, оскільки процедура обробки кожного пікселя зображення займає певний час.

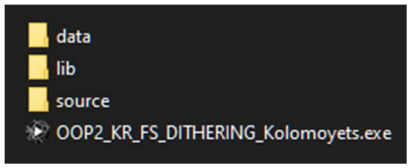

Рис. 5. Відображення папки з портативною версією програми

На рис.6. показано результати застосування програми у випадку використання для формули 1 значення фактору зі значенням рівним 1.

Рис. 6. Відображення результату виконання з фактором 1

Наступний рис. 7. відображає результати застосування програми у випадку використання для формули 1 значення фактору зі значенням рівним 4.

Рис. 7. Відображення результату виконання з фактором 4

Таким чином, ми бачимо, що відображення обробки графічного зображення із використанням застосування алгоритму Флойда-Стейнберга є ефективним і результати залежать від показника значення фактору.

5. Висновок

У роботі досліджено застосування алгоритму Флойда-Стейнберга для роботи з графічними зображеннями.

Аналіз даних для різних алгоритмів засвідчив, що всі алгоритми дизерингу на основі дифузії помилки квантизації розподіляють значення помилки квантизації на сусідні пікселі, але виконують це по різному. Алгоритми, які працюють на основі квантизації, але не використовують значення похибки квантизації, наприклад Ordered Dithering [9], передбачають для кожного пікселя зміну свого значення кольору на найближчий у зменшеній квантизованій палітрі. Результати дослідження засвідчили ефективність та доцільність застосування спеціальних алгоритмів для генерування GIF-зображень, де обмежена палітра кольорів та виконується стискання зображення.

З наявних на сьогоднішній час алгоритмів згладжування зображень найбільш часто використовується алгоритм Флойда-Стейнберга, який зараз застосовується у сервісах для створення GIF-зображень та анімацій.

Отримані результати засвідчують ефективність та надійність використання алгоритму Флойда-Стейнберга для обробки зображень. Інтерфейс програми передбачає мінімальну кількість дій для виконання обробки зображень.

Список літератури

1. Floyd–Steinberg dithering – [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Floyd%E2%80%93Steinberg_dithering

2. Floyd–Steinberg dithering – [Електронний ресурс] – Режим доступу: https://www.visgraf.impa.br/Courses/ip00/proj/Dithering1/floyd_steinberg_dithering.html

3. Error Diffusion dithering – Computerphile – [Електронний ресурс] - Режим доступу: https://www.youtube.com/watch?v=ico4fJfohMQ

4. Середовище розробки Processing *– [Електронний ресурс] - Режим доступу:* processing.org

5. Shakib, J., Muqri, M., Leveraging the Power of Java in the Enterprise, American Society for Engineering Education, AC 2010-1701.

6. Dibble, P., Real-Time Java Platform Programming, Sun Microsystems Press, Prentice-Hall, June 2008.

7. Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java. К.; Диалектика, 2019.-848 с.

8. Palmer G., Technical Java - Developing Scientific and Engineering Applications, Prentice Hall, 2003.

9. Paul Jensen. An Edge-Preserving Inverse Halftoning Algorithm for Ordered Dithered Images – [Електронний ресурс] – Режим доступу: https://core.ac.uk/download/236185916.pdf

References

1. Floyd–Steinberg dithering – [Elektronnyi resurs] – Rezhym dostupu: https://en.wikipedia.org/wiki/Floyd%E2%80%93Steinberg_dithering

2. Floyd–Steinberg dithering – [Elektronnyi resurs] – Rezhym dostupu: https://www.visgraf.impa.br/Courses/ip00/proj/Dithering1/floyd_steinberg_dithering.html

3. Error Diffusion dithering – Computerphile – [Elektronnyi resurs] – Rezhym dostupu: https://www.youtube.com/watch?v=ico4fJfohMQ

4. Seredovyshche rozrobky Processing – [Elektronnyi resurs] – Rezhym dostupu: processing.org

5. Shakib, J., Muqri, M., Leveraging the Power of Java in the Enterprise, American Society for Engineering Education, AC 2010-1701.

6. Dibble, P., Real-Time Java Platform Programming, Sun Microsystems Press, Prentice-Hall, June 2008.

7. Robert Sedzhvik, Kevin Ujejn. Algoritmy na Java. K.; Dialektika, 2019.-848 s.

8. Palmer G., Technical Java - Developing Scientific and Engineering Applications, Prentice Hall, 2003.

9. Paul Jensen. An Edge-Preserving Inverse Halftoning Algorithm for Ordered Dithered Images – [Elektronnyi resurs] – Rezhym dostupu: https://core.ac.uk/download/236185916.pdf