Обробка зображень і мультимедіа

Олег Гутік

Лекція 19: Стиснення зображень, XIII. Прогресуюче стиснення зображень

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Більшість сучасних методів стиснення зображень є прогресуючими (поступальними) або вони легко робляться такими. Це особливо важливо, якщо стиснене зображення передається каналами зв'язку, а потім декодується та спостерігається одержувачем в режимі реального часу (наприклад, це робиться Internet-браузерами). Коли таке зображення приймається та розтискається, декодер здатний дуже швидко показати всю картинку у форматі з низькою якістю, а потім поступово покращувати якість у міру прийому решти стисненого зображення та його декодування. Користувач, спостерігаючи на екрані розвиток образу, може розпізнати всі особливості цього зображення після декодування всього 5–10% його розміру.

Це можна порівняти зі стисненням растрово-сканованих зображень. Коли зображення сканується та стискається по рядках зверху вниз, а потім декодується послідовно, користувач зазвичай не може багато дізнатися про нього, спостерігаючи на дисплеї лише 5–10% від усього зображення. Оскільки картинки будуть розглядатися людьми, то прогресуюче стиснення є кращим, навіть якщо воно, загалом, працює повільніше та має меншу ефективність, ніж непрогресуюче стиснення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Для того, щоб краще зрозуміти прогресуюче стиснення, уявіть собі, що кодер спочатку стискає найважливішу частину зображення та поміщає його у файл, потім стискується менш важлива інформація та додається в стислий файл і т.д. Це також пояснює, чому втрату інформації при прогресуючому стисненні можна контролювати природним чином: досить зупинити стиснення у певній точці. Користувач може змінювати частку опущеної інформації за допомогою параметра, який повідомляє кодера, як довго слід продовжувати процес стиснення. Чим раніше станеться ця зупинка, тим вищим буде ступінь стиснення та тим більшим буде розмір втраченої інформації.

Інша перевага прогресуючого стиснення стає незаперечною, коли стислий файл повинен розтискатися багато разів і щоразу з різною роздільною здатністю. Декодер може в кожному випадку зупинити процес декодування, коли зображення досягло роздільної здатності, доступної даному пристрою.

Ми вже згадували про прогресуюче стиснення зображень у зв'язку з алгоритмом JPEG. Цей алгоритм використовує DCT для поділу образу на просторові частоти та починає стиснення з низькочастотних компонентів. Тому декодер може швидко показати відповідну їм грубу картинку. Високочастотні компоненти містять деталі зображення.

Стиснення зображень. Прогресуюче стиснення зображень

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням.

Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Крім того, корисно уявляти собі прогресуюче декодування як процес покращення якості зображення у часі. Це можна зробити трьома шляхами:

(1) Закодувати просторові частоти зображення прогресуючим чином. Спостерігач, що стежить за декодуванням, побачить таке зображення змінюваним від розмитого до різкого. Методи, які працюють подібним чином, мають середню швидкість кодування та повільну швидкість декодування. Цей тип стиснення часто називається *прогресуючим за співвідношенням сигнал/шум* або *прогресуючим за якістю зображення*.

(2) Почати з чорно-білого напівтонового зображення, а потім додати колір і тіні. При декодуванні глядач із самого початку побачить зображення з усіма деталями, які покращуватимуться в міру додавання кольорів та фарб. Метод векторного квантування застосовує цей принцип компресії. Такі алгоритми відрізняються повільним кодуванням та швидким декодуванням.

(3) Кодувати зображення шарами, де ранні шари складаються з декількох пікселів великої роздільної здатності. Спостерігач помітить поступову деталізацію картинки у часі під час її декодування. Такий метод додає нові деталі в міру розтискання файлу. Цей шлях прогресуючого стиснення називається *пірамідальним* чи *ієрархічним* кодуванням. Більшість методів стиснення використовують цей принцип, і в цій лекції обговорюються загальні ідеї здійснення пірамідального кодування.

Стиснення зображень. Прогресуюче стиснення зображень

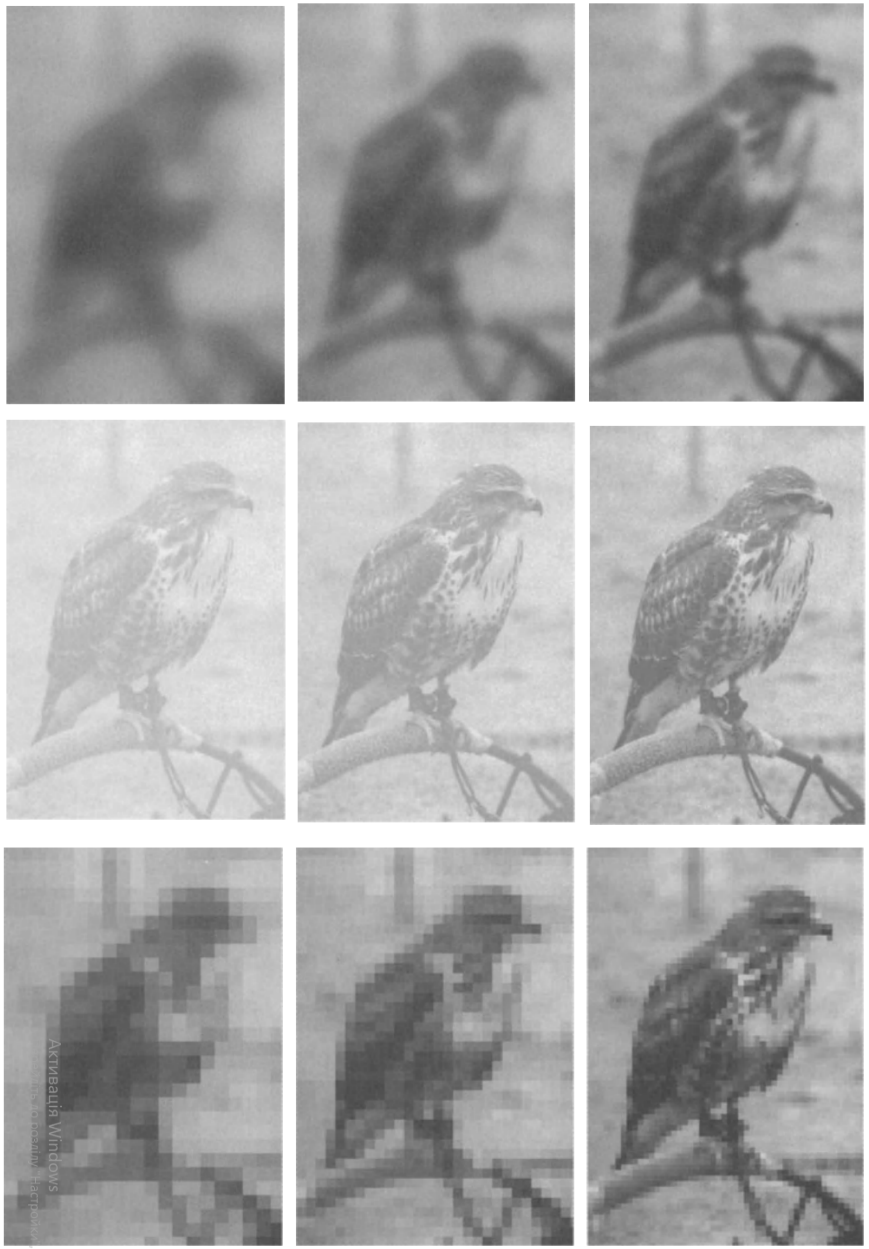


Рис. ілюструє три принципи прогресуючого стиснення, перераховані вище.

Стиснення зображень. Прогресуюче стиснення зображень

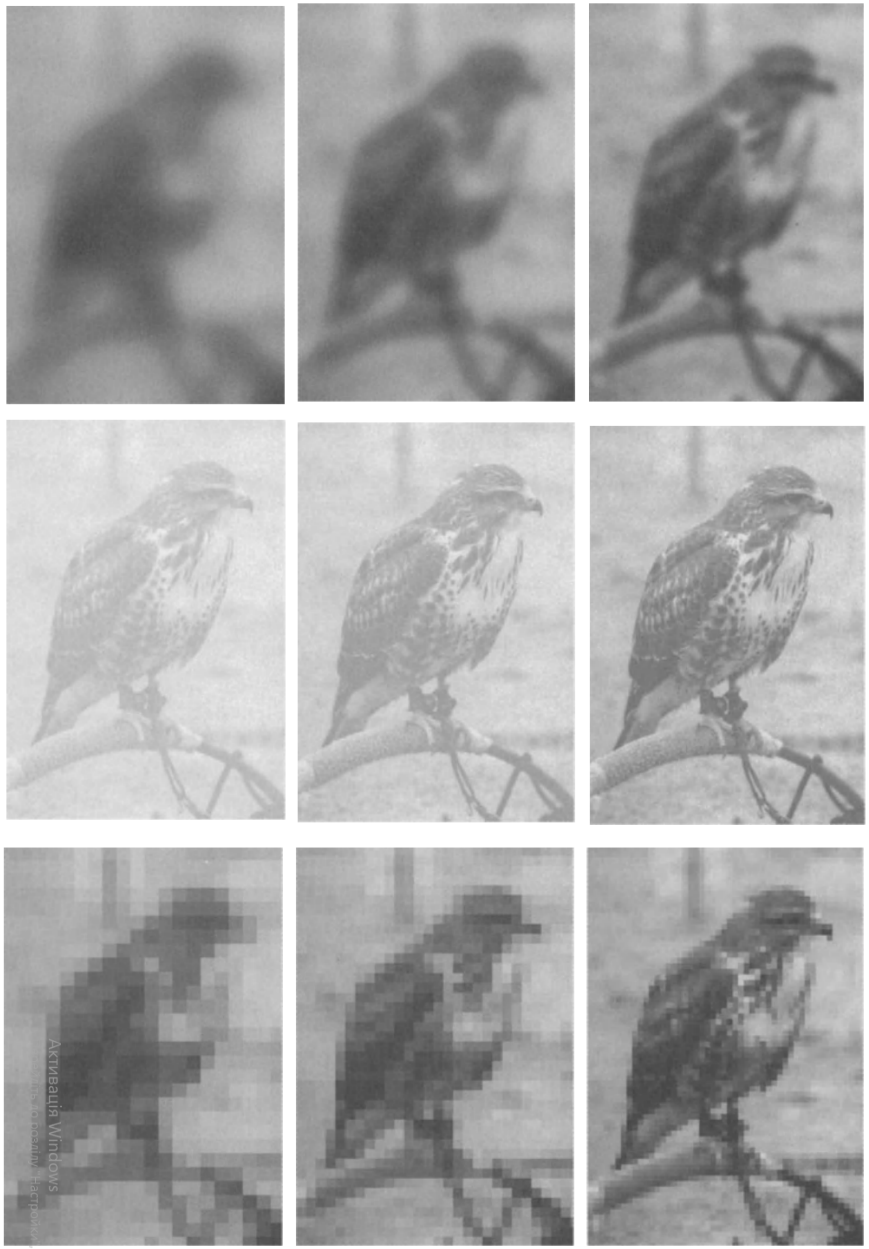


Рис. ілюструє три принципи прогресуючого стиснення, перераховані вище.

Стиснення зображень. Прогресуюче стиснення зображень

Вони контрастують із рис., на якому зображені етапи послідовного декодування.

Вони контрастують із рис., на якому зображені етапи послідовного декодування.

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$
\begin{aligned}
4^0 + 4^1 + \cdots + 4^{n-1} + 4^n &= \frac{4^{n+1}-1}{4-1} = \\
&= \frac{4^{n+1}-1}{3} \approx \\
&\approx 1.33 \cdot 4^n \approx \\
&\approx 1.33 \cdot (2 \cdot 2)^n,
\end{aligned}
$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів.

Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів.

Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} = \frac{4^{n+1}-1}{3} \approx \approx 1.33 \cdot 4^n \approx \approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} =$$

$$= \frac{4^{n+1}-1}{3} \approx$$

$$\approx 1.33 \cdot 4^n \approx$$

$$\approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} =$$
$$= \frac{4^{n+1}-1}{3} \approx$$
$$\approx 1.33 \cdot 4^n \approx$$
$$\approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} =$$
$$= \frac{4^{n+1}-1}{3} \approx$$
$$\approx 1.33 \cdot 4^n \approx$$
$$\approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} =$$
$$= \frac{4^{n+1}-1}{3} \approx$$
$$\approx 1.33 \cdot 4^n \approx$$
$$\approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$4^0 + 4^1 + \cdots + 4^{n-1} + 4^n = \frac{4^{n+1}-1}{4-1} =$$

$$= \frac{4^{n+1}-1}{3} \approx$$

$$\approx 1.33 \cdot 4^n \approx$$

$$\approx 1.33 \cdot (2 \cdot 2)^n,$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$
\begin{aligned}
4^0 + 4^1 + \cdots + 4^{n-1} + 4^n &= \frac{4^{n+1} - 1}{4 - 1} = \\
&= \frac{4^{n+1} - 1}{3} \approx \\
&\approx 1.33 \cdot 4^n \approx \\
&\approx 1.33 \cdot (2 \cdot 2)^n,
\end{aligned}
$$

що на 33% більше розміру вихідної кількості пікселів!

Стиснення зображень. Прогресуюче стиснення зображень

Припустимо, що зображення складається з $2^n \times 2^2 = 4^n$ пікселів. Найпростіший метод прогресуючого стиснення, який спадає на думку, полягає у обчисленні кожного пікселя шару $i-1$ у вигляді середньої групи 2×2 пікселів шару i. Отже, шар n — це вихідне зображення (4^n пікселів розміру 1), шар $n-1$ складається з $2^{n-1} \times 2^{n-1} = 4^{n-1}$ пікселів розміру 2×2, і т.д. до шару $4^{n-n} = 1$ з одного великого пікселя розміру 4^n, який представляє все зображення. Якщо зображення не надто велике, то всі шари можна зберегти в пам'яті комп'ютера. Потім шари записуються у файл у зворотному порядку, починаючи з шару 1. Єдиний піксель шару 1 є "батьком" чотирьох пікселів шару 2, кожен з яких породжує по 4 пікселі шару 3 і т.д. Цей процес породжує прогресуючий файл образу, але без стиснення, оскільки загальна кількість пікселів цієї піраміди дорівнює

$$
\begin{aligned}
4^0 + 4^1 + \cdots + 4^{n-1} + 4^n &= \frac{4^{n+1}-1}{4-1} = \\
&= \frac{4^{n+1}-1}{3} \approx \\
&\approx 1.33 \cdot 4^n \approx \\
&\approx 1.33 \cdot (2 \cdot 2)^n,
\end{aligned}
$$

що на 33% більше розміру вихідної кількості пікселів!

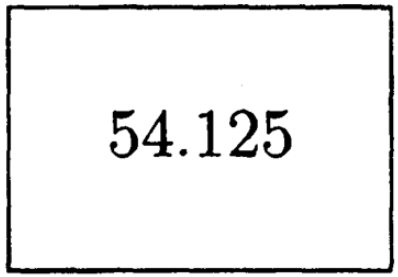
Стиснення зображень. Прогресуюче стиснення зображень

Простий метод зведення кількості пікселів піраміди до вихідного числа 4^n полягає у збереженні лише трьох пікселів шару i з чотирьох та в обчисленні четвертого за допомогою трьох його "братів" і одного батьківського пікселя цієї групи з попереднього шару $i-1$.

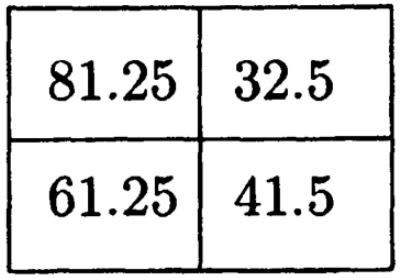
Простий метод зведення кількості пікселів піраміди до вихідного числа 4^n полягає у збереженні лише трьох пікселів шару i з чотирьох та в обчисленні четвертого за допомогою трьох його "братів" і одного батьківського пікселя цієї групи з попереднього шару $i-1$.

Простий метод зведення кількості пікселів піраміди до вихідного числа 4^n полягає у збереженні лише трьох пікселів шару i з чотирьох та в обчисленні четвертого за допомогою трьох його "братів" і одного батьківського пікселя цієї групи з попереднього шару $i-1$.

Простий метод зведення кількості пікселів піраміди до вихідного числа 4^n полягає у збереженні лише трьох пікселів шару i з чотирьох та в обчисленні четвертого за допомогою трьох його "братів" і одного батьківського пікселя цієї групи з попереднього шару $i-1$.



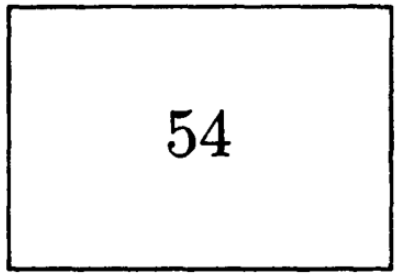
(a)



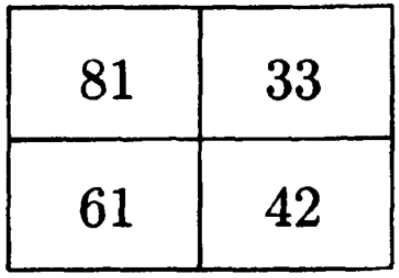
(b)

90	72	58	33
140	23	21	18
100	70	72	18
16	59	44	32

(c)



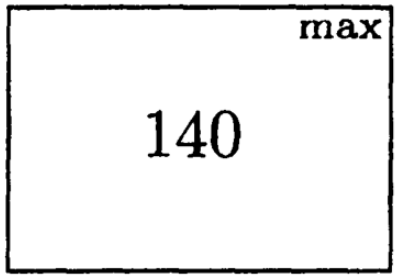
(d)



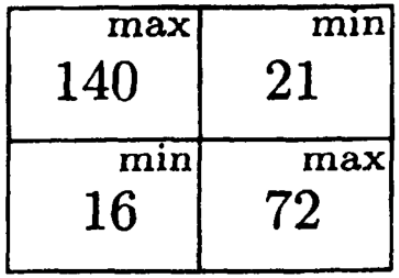
(e)

90	72	58	33
140	23	21	18
100	70	72	18
16	59	44	32

(f)



(g)



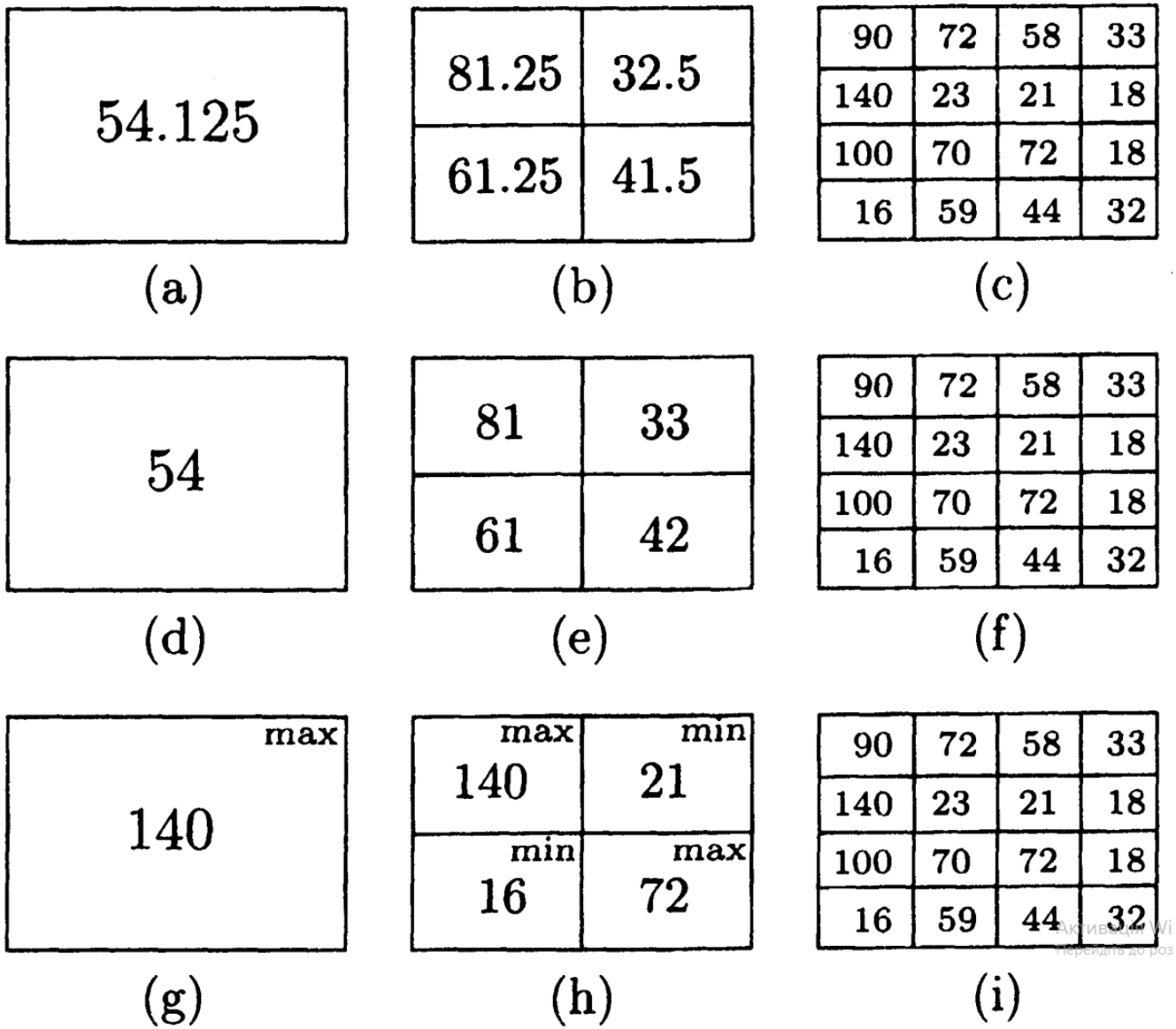
(h)

90	72	58	33
140	23	21	18
100	70	72	18
16	59	44	32

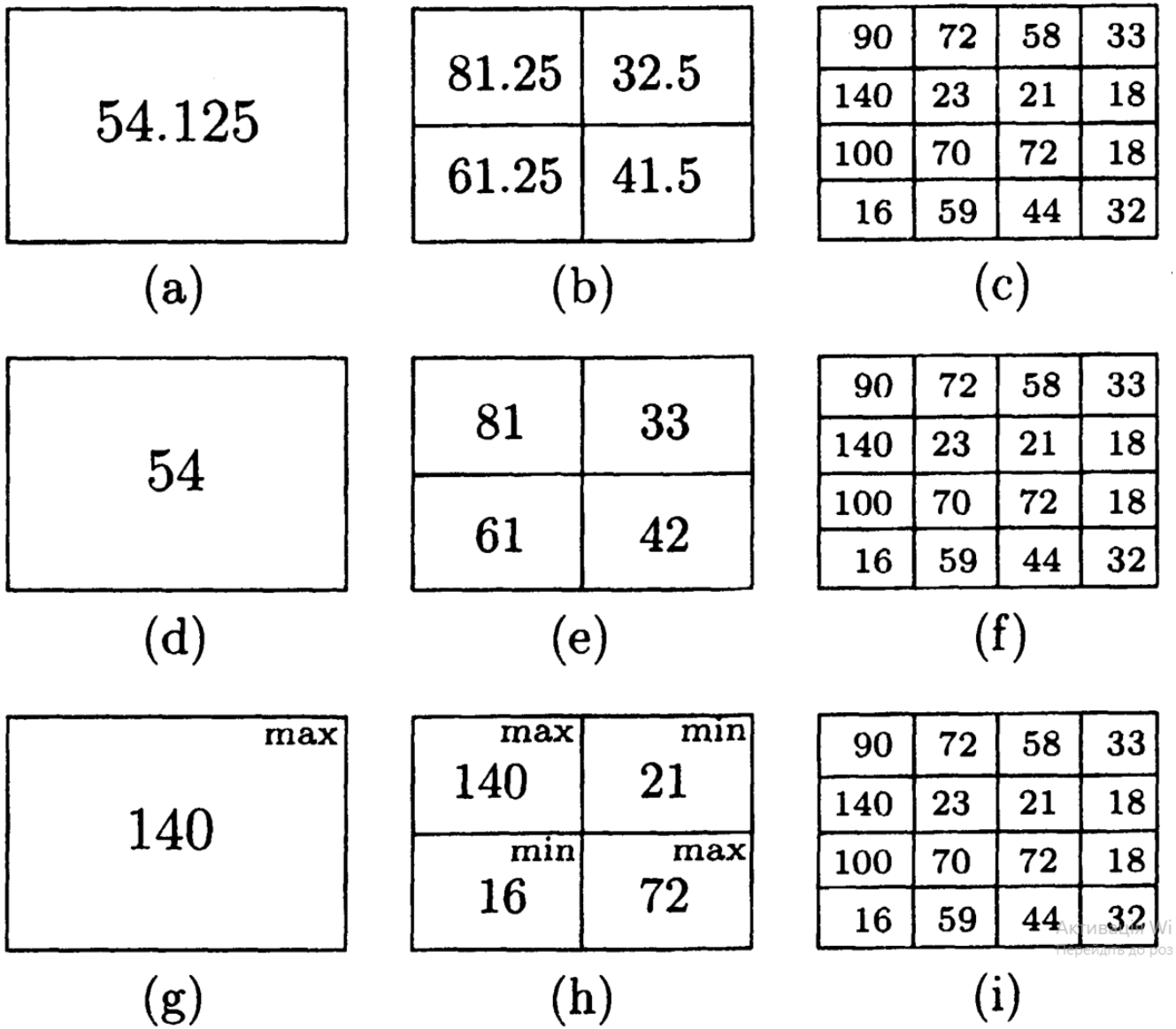
(i)

На рис. (c) викладено зображення розміру 4×4 яке є третім шаром прогресуючого стиснення.

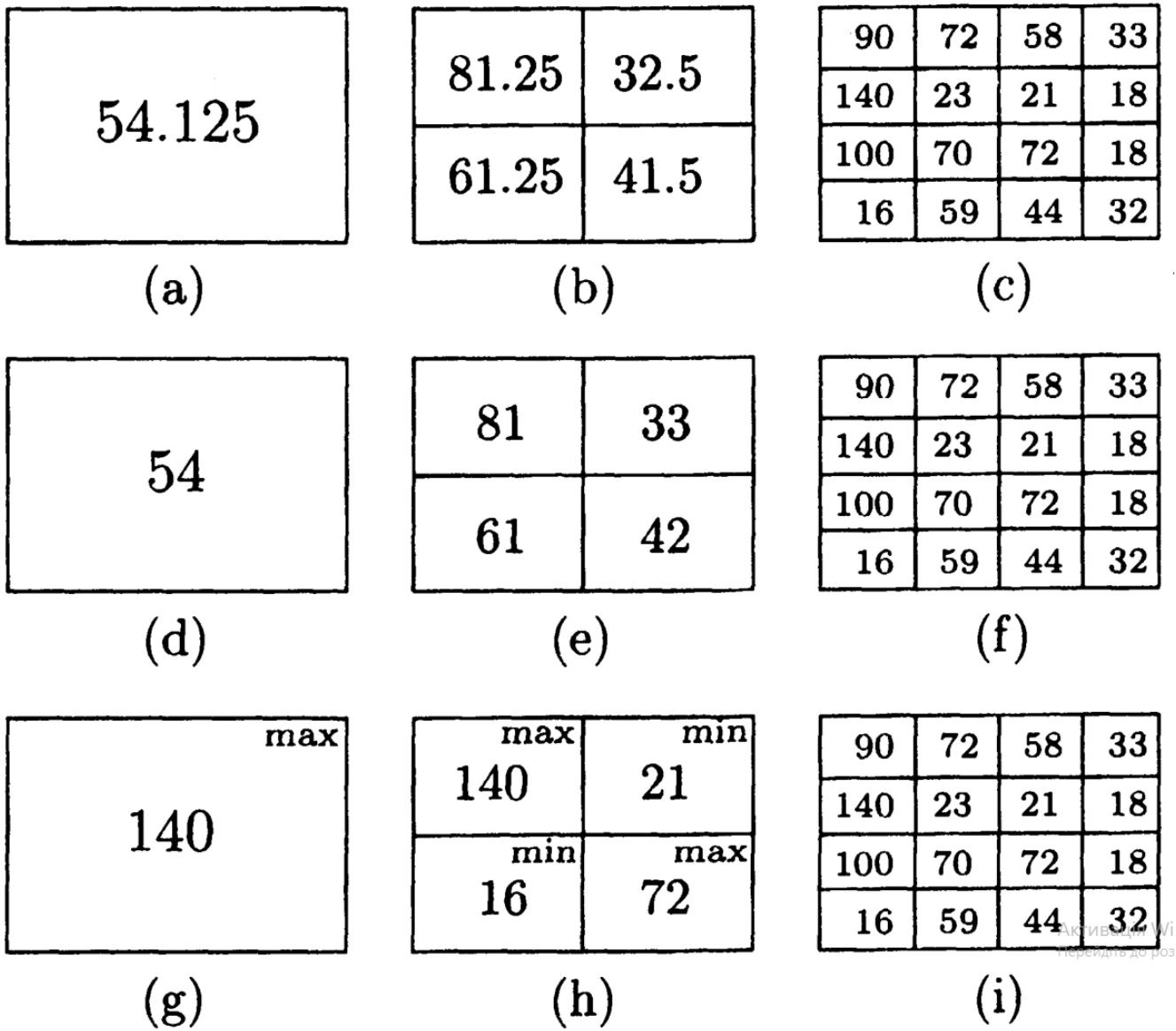
Стиснення зображень. Прогресуюче стиснення зображень



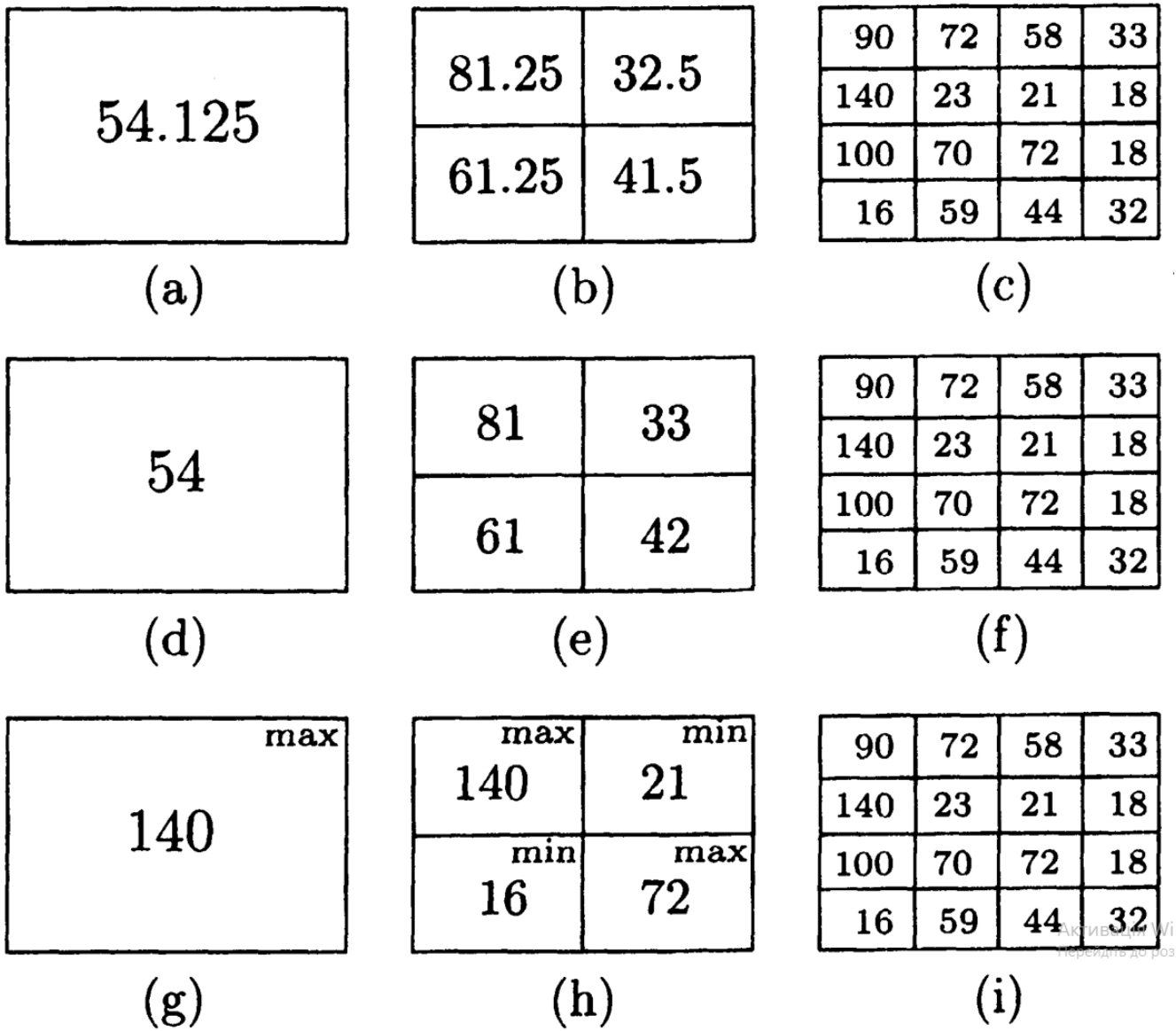
На рис. (c) викладено зображення розміру 4×4 яке є третім шаром прогресуючого стиснення.



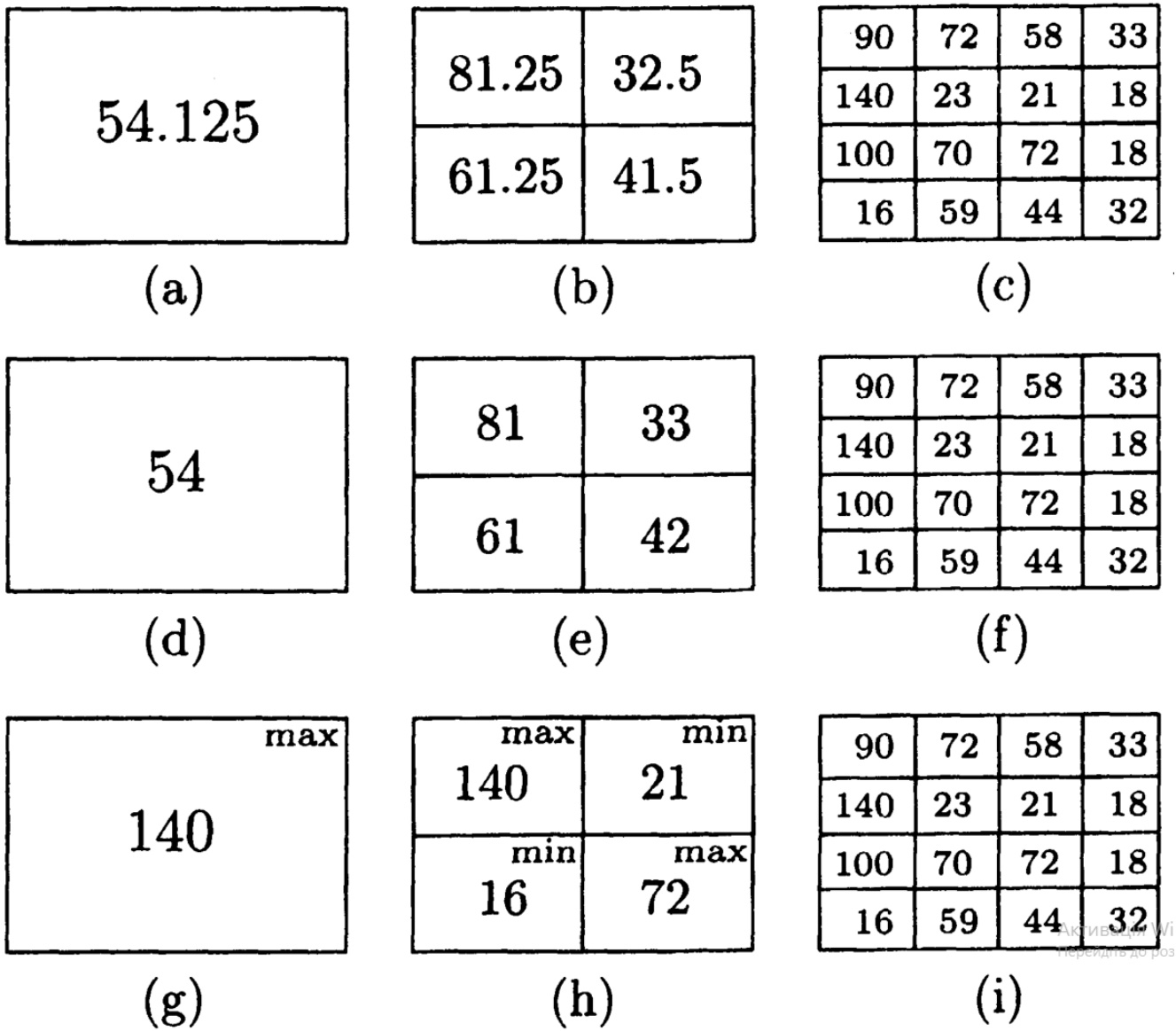
Другий шар показано на рис. (b), де, наприклад, піксель 81.25 є середнім значенням чотирьох пікселів 90, 72, 140 та 23 з третього шару.



Другий шар показано на рис. (b), де, наприклад, піксель 81.25 є середнім значенням чотирьох пікселів 90, 72, 140 та 23 з третього шару.



Єдиний піксель першого шару наведено на рис.(а).



Єдиний піксель першого шару наведено на рис.(a).

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснений файл міститиме лише числа

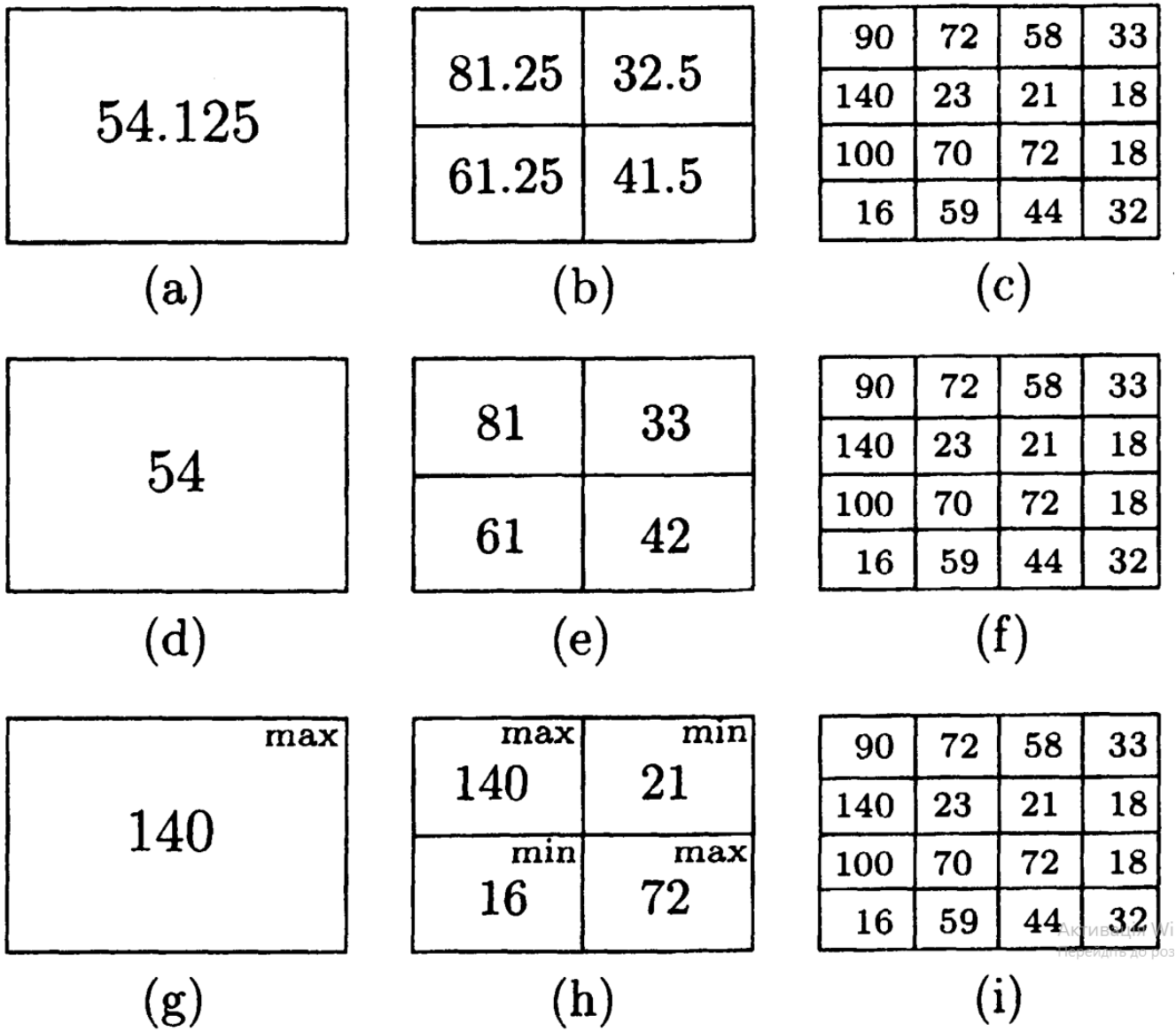
$$54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16$$

(звичайно, правильно закодовані) з яких легко відновити пікселі, які відсутні. Наприклад, відсутній піксель 81.25 обчислюється з рівняння

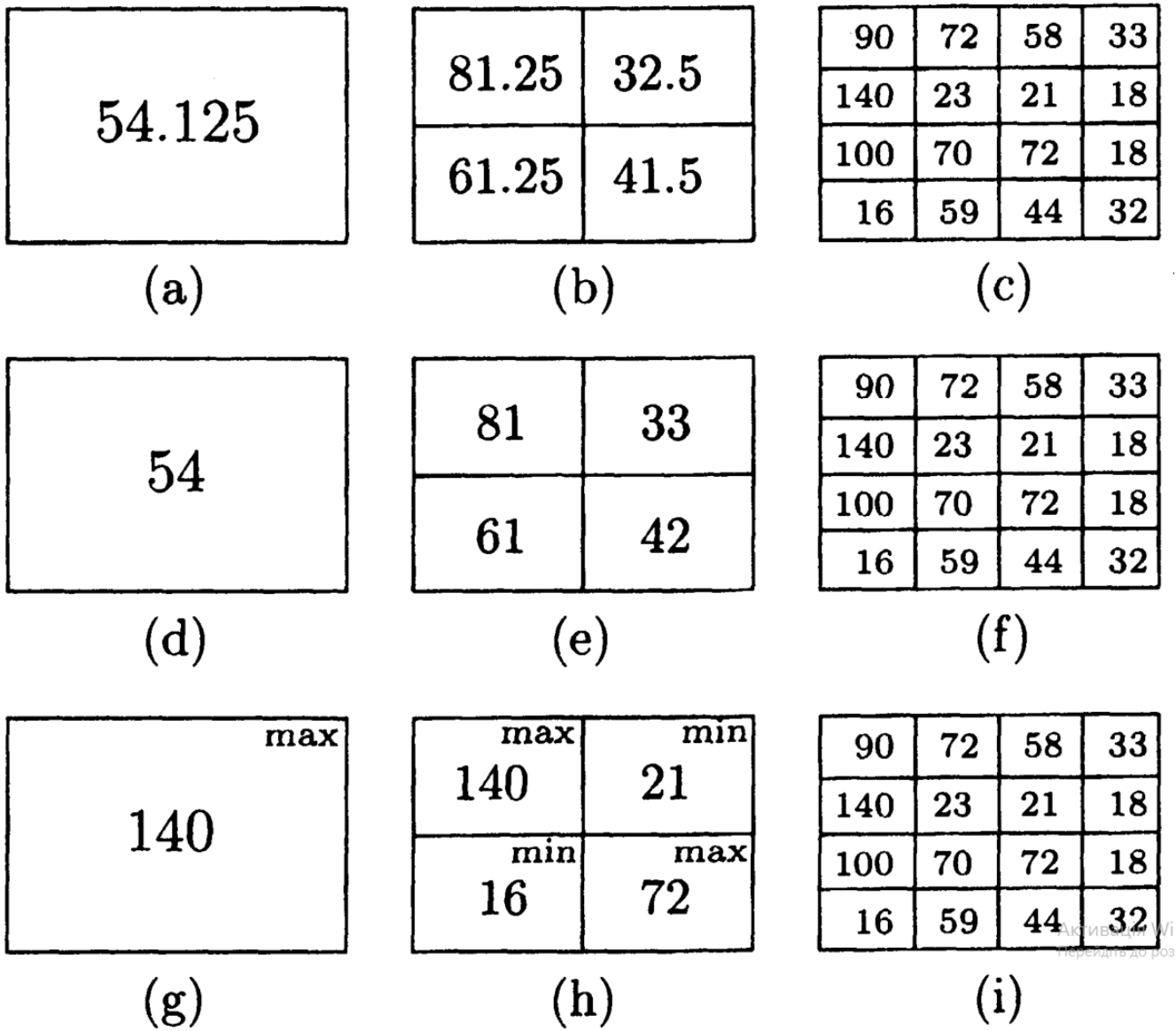
$$\frac{x + 32.5 + 41.5 + 61.25}{4} = 54.125.$$

Проблема у тому, що середнє значення цілих чисел може бути нецілим числом. Якщо ми хочемо, щоб пікселі залишалися цілими, то доведеться або змиритися з втратою точності, або зберігати дедалі більші цілі числа. Наприклад, якщо вихідні пікселі мають 8 розрядів, то додавання чотирьох 8-ми бітових чисел дає 10-бітове число. Розподіл цього числа на 4 для обчислення середнього та округлення до цілого знову призводить до 8 розрядів, але з можливою втратою точності. Якщо втрата точності небажана, то можна уявити наш піксель другого шару у вигляді 10-ти бітових чисел, а піксель першого шару — за допомогою 12 разів рядів.

Стиснення зображень. Прогресуюче стиснення зображень



На рис. (d),(e),(f) показаний результат заокруглення, у якому відбувається деяка втрата інформації.



На рис. (d),(e),(f) показаний результат заокруглення, у якому відбувається деяка втрата інформації.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Вмістом стисненого файлу буде послідовність

$$54, 33, 42, 61, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16.$$

Перший відсутній піксель 81 другого шару можна обчислити з рівняння

$$\frac{x + 33 + 42 + 61}{4} = 54.$$

Вийде число 80 з невеликою помилкою.

Кращий розв'язок полягає у використанні батька групи при обчисленні його чотирьох нащадків. Це можна робити, обчислюючи різницю між батьком та його нащадками та записуючи цю різницю (після відповідного кодування) у шар i стисненого файлу. Декодер відновлює різницю і використовує батька з шару $i - 1$ для обчислення значень чотирьох пікселів. Для кодування різниць можна використовувати метод Гаффмана або арифметичне кодування. Якщо всі шари обчислені та перебувають у пам'яті, то можна знайти статистичний розподіл цих різниць, який можна використовувати для досягнення найкращого статистичного стиснення.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a - b, \quad l = b - c \quad \text{і} \quad m = c - d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a - b, \quad l = b - c \quad \text{і} \quad m = c - d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a - b, \quad l = b - c \quad \text{і} \quad m = c - d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a - b, \quad l = b - c \quad \text{і} \quad m = c - d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Якщо в пам'яті немає всіх шарів, то можна застосувати просту адаптивну модель. Вона починається присвоєнням лічильника 1 кожному значенню різниці (щоб уникнути проблеми нульової ймовірності). Після обчислення чергової різниці, їй надається деяка ймовірність, і вона кодується, виходячи з її лічильника. Після цього лічильник оновлюється. Непогано робити збільшення лічильника не на одиницю, але на більше число. Тоді вихідні одиничні значення лічильників швидко стануть незначними.

Деякого поліпшення можна досягти, якщо використовувати батьківський піксель при обчисленні трьох нащадків, а потім цих трьох нащадків і батька використовувати при обчисленні четвертого пікселя цієї групи. Якщо чотири пікселі групи дорівнюють a, b, c та d, то їхнє середнє дорівнює

$$v = \frac{a+b+c+d}{4}.$$

Це середнє стає частиною шару $i-1$, а шар i може містити лише три різниці

$$k = a-b, \quad l = b-c \quad \text{і} \quad m = c-d.$$

Після того, як декодер прочитає і декодує ці три різниці, він може обчислити всі чотири пікселі групи за допомогою значення v попереднього шару. Обчислення v за допомогою поділу на 4 все ще означає втрату двох бітів, але цю двобітову величину можна виділити до поділу і кодувати її окремо за трьома різницями.

Стиснення зображень. Прогресуюче стиснення зображень

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

Батьківський піксель не повинен дорівнювати середньому значенню групи. Можна, наприклад, вибрати як батьківський максимальний (або мінімальний) піксель групи. Перевага такого методу полягає в тому, що батьківський піксель дорівнює одному з його нащадків. Кодер просто закодує три пікселі групи, а декодер декодує три пікселі (або їхні різниці) і доповнить групу четвертим батьківським пікселем. При кодуванні послідовних груп шару, кодер повинен вибирати по черзі максимальне або мінімальне значення в якості батька, оскільки вибір тільки одного екстремуму породить шари, що стають темнішими або світлішими.

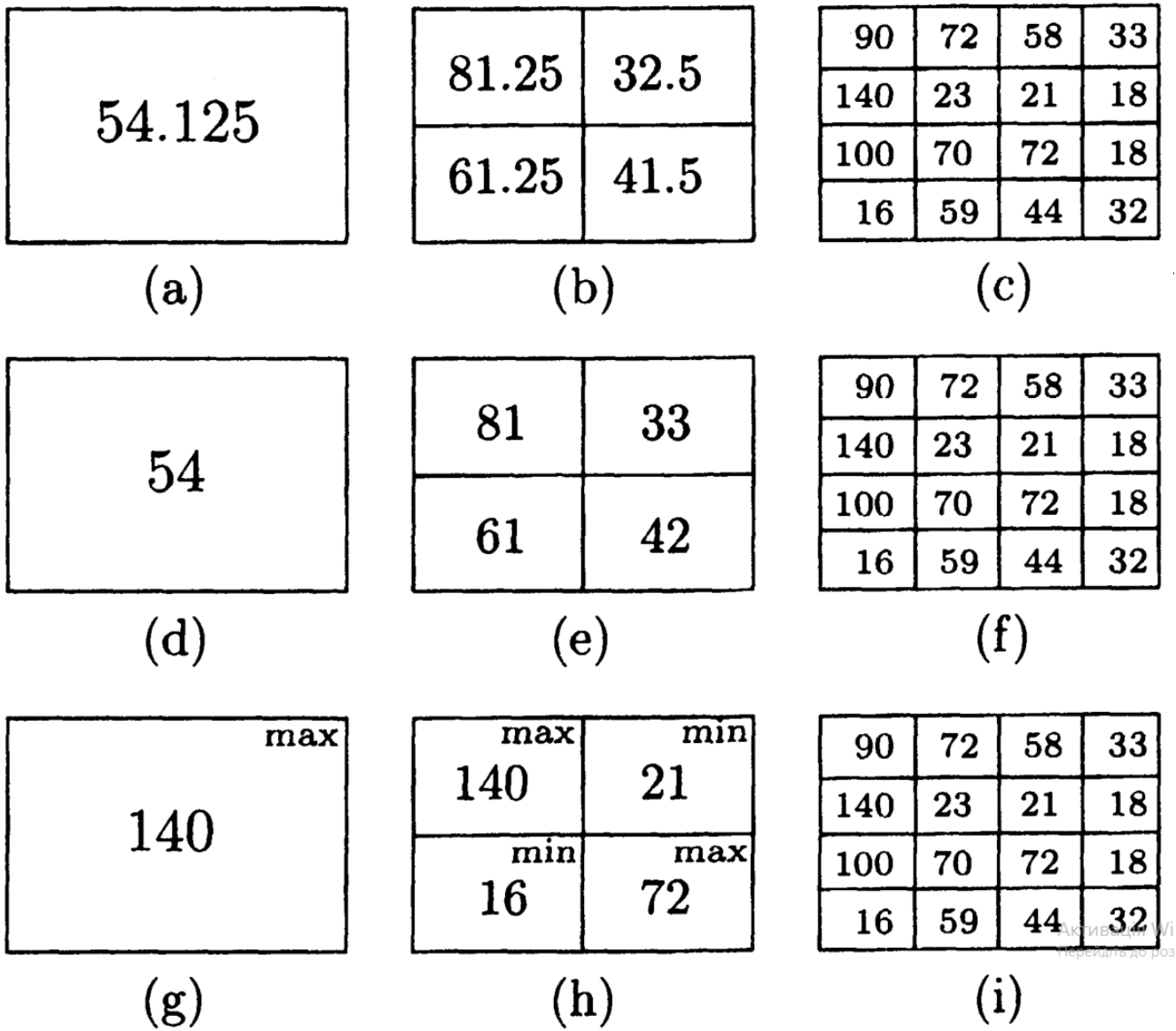


Рис. (g),(h),(i) показує три шари для цього випадку.

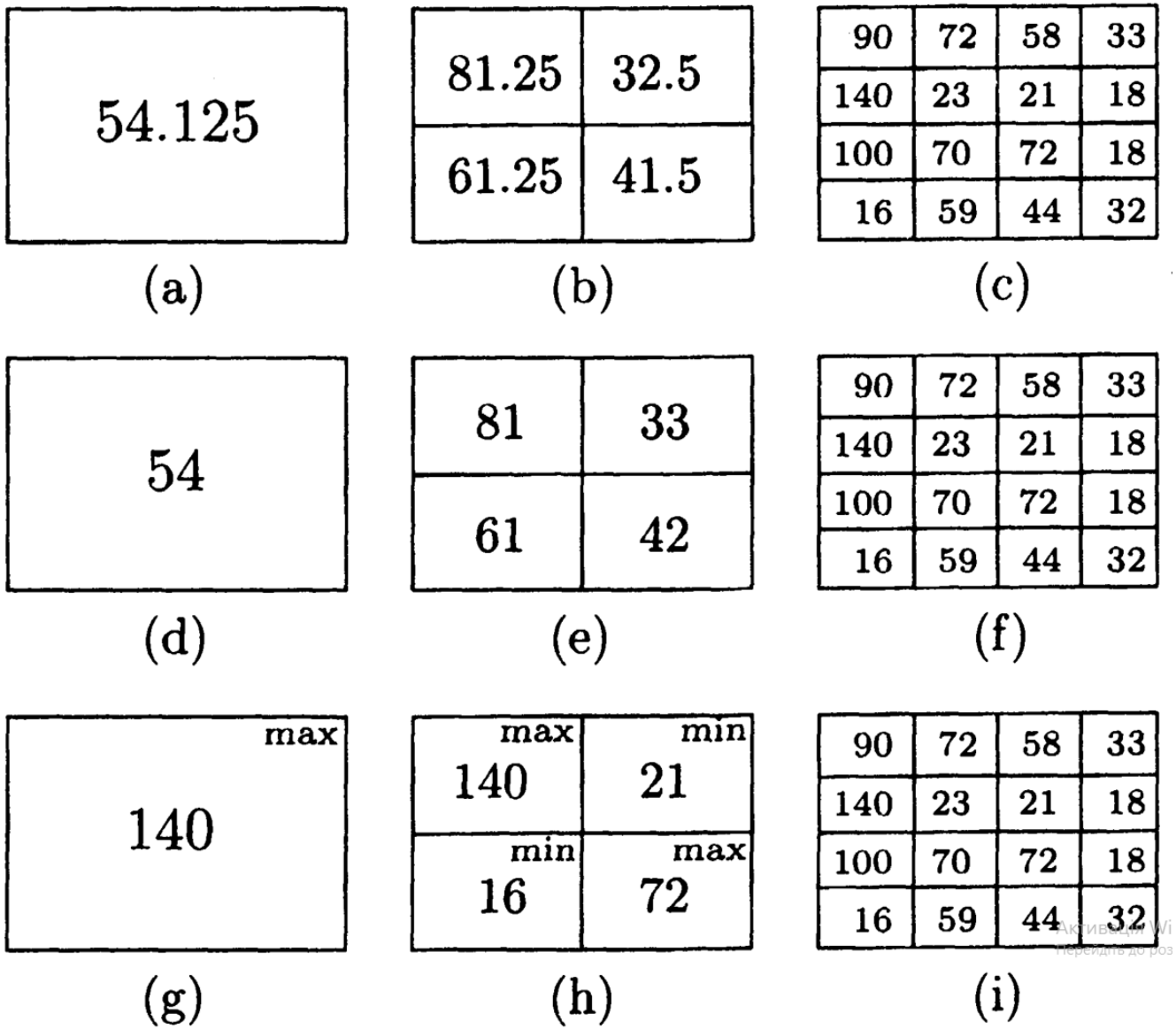


Рис. (g),(h),(i) показує три шари для цього випадку.

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

У стиснутому файлі будуть записані числа

$$140, (0), 21, 72, 16, (3), 90, 72, 23, (3), 58, 33, 18, (0), 18, 32, 44, (3), 100, 70, 59,$$

де число в дужках має довжину 2 біти. Воно говорить про те, в який квадрант помістити батьківський піксель. Зауважимо, що квадранти занумеровані так:

$$\begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}.$$

Вибір *медіани* групи відбувається дещо повільніше знаходження максимуму або мінімуму, але покращує динаміку прояву шарів при прогресуючій декомпресії. За означенням, *медіаною послідовності чисел* $(a_1, a_2, \ldots, a_n)$ називається такий елемент a, що половина (або майже половина) елементів послідовності менше за a, а інша половина — більша за a. Якщо чотири елементи групи пікселів задовольняють нерівності $a < b < c < d$, то медіаною буде число b або c. Основна перевага вибору медіани для батьківського пікселя полягає в тому, що це дозволить згладити великі різниці значень пікселів. У групі 1, 2, 3, 100 вибір в якості батька числа 2 або 3 є більш характерним для цієї групи, ніж вибір середнього числа. Знаходження медіани вимагає двох порівнянь, а обчислення середнього знадобиться розподіл на 4 (чи правий зсув).

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Якщо використовується медіана при кодуванні шару $i-1$, то три пікселя, що залишилися, можна кодувати в шарі i за допомогою їхніх різниць, за якими слідує 2-х бітовий код, що повідомляє, який з чотирьох був батьківським. Інша невелика перевага використання медіани полягає в тому, що коли декодер читає цей 2-х бітовий код, йому стає відомо, скільки з трьох пікселів більше медіани, а скільки — менше. Наприклад, якщо цей код говорить, що один піксель менше медіани, а два інші — більше, і прочитаний піксель менше медіани, то декодер точно знає, що пікселі, що залишилися, будуть більше медіани. Отримана інформація змінює розподіл різниць пікселів, і ця перевага може бути реалізована за допомогою трьох таблиць лічильників для визначення ймовірностей кодування різниць. Одна таблиця використовується при кодуванні пікселів за умови, що декодер знає, що вони більші за медіану. Інша таблиця застосовується для кодування, якщо відомо, що піксель менше медіани. А третя таблиця працює, якщо декодер не знає заздалегідь їхнє співвідношення з медіаною. Ця обставина покращує стиснення на кілька відсотків і показує, як додавання нових ознак метод стиснення дає загальне зменшення стисненого файлу.

Дякую за увагу!