УДК 004.931 doi: 10.15421/321910

С.М. Єфремов, Т.А. Зайцева
Дніпровський національний університет імені Олеся Гончара

ОПТИМІЗАЦІЯ РОЗРАХУНКУ ГІСТОГРАМИ НАПРЯМЛЕНИХ ГРАДІЄНТІВ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕННЯ

В роботі розглядається оптимізація розрахунку гістограми напрямлених градієнтів для розпізнавання змісту зображення із застосуванням інтерполяції та алгоритму оптимізації швидкості розрахунків. Виконана програмна реалізація алгоритму, та на тестових прикладах проведена оцінка роботи програми.

Ключові слова: оптимізація, інтерполяція, HOG дескриптор, комп'ютерний зір, розпізнавання образів, інтегральне представлення зображень, гістограма напрямлених градієнтів.

В работе рассматривается оптимизация расчета гистограммы направленных градиентов для распознавания содержания изображения с использованием интерполяции и алгоритма оптимизации скорости расчетов. Выполнена программная реализация алгоритма и на тестовом примере продемонстрированы результаты его работы.

Ключевые слова: оптимизация, интерполяция, HOG дескриптор, компьютерное зрение, распознавание образов, интегральное представление изображений, гистограмма направленных градиентов.

The possibilities of the optimization of histogram of oriented gradients calculations for solving image content recognition problems described based on the 48 × 48 pixels size image example. The algorithm doesn't change in regards to the input data and suits for the histogram of the oriented gradients calculation based on any image. The algorithm idea is taken from the work of Soojin Kim and Kyeongsoon Cho [1], which is the modification of original HOG descriptor algorithm presented by Navneet Dalal and Bill Triggs [3] aimed at optimization of the calculation speed without loosing accuracy during image content recognition using HOG descriptor to generate the set of features of the image content. The algorithm is described in detail in the next sequence of actions. 1) Original HOG feature calculation. We use it as the first step since the algorithm is the optimized version of the original HOG feature calculation. 2) Solving the aliasing problem and accuracy improvement by using the interpolation technique during the HOG feature calculation process. We use one of the normalization schemes, applying interpolation as the next calculation step. 3) Solving the redundant operations and calculation speed problems by using cell-based operations and applying from one to four described types to the cells, based on which depends the detection window cell calculation. The types are applied to cell based on the blocks intersection containing the cell.

The computer program has been developed according to the selected optimized algorithm of HOG feature calculation. It was used during the image content features description and learning process and in the further computer vision research. The software implementation of the algorithm takes into account the capabilities of modern computer technology, Javascript programming language and modern needs of the image content recognition calculation speed and accuracy.

The implementation of computer program logic is shown in the script examples, utilizing modularity and parallel calculation as the strong sides of Javascript, further improving HOG feature calculation speed.

Keywords: optimization, interpolation, HOG feature descriptor, computer vision, pattern recognition, integral image representation, histogram of oriented gradients.

Вступ. Сьогодні серед інтелектуальних систем значного поширення набули системи комп'ютерного зору, в яких актуальними є задачі розпізнавання зображень. На даний час розроблено ряд методів і алгоритмів синтезу та аналізу зображень, які ґрунтуються на різних теоретичних засадах. Гістограма напрямлених градієнтів (HOG) є сучасним методом опису характеристик змісту зображення, дескриптором, що активно використовується у сфері комп'ютерного зору, дозволяючи отримати набір чітких ознак образів на зображенні, що в свою чергу дозволяє навчання та використання нейронної сітки з ціллю близької до людської точності розпізнавання змісту зображення. Даний метод надає більшу точність та є відносно простим у реалізації відносно більшості аналогів [2], а саме тому є перспективним та широко застосовуваним у сфері комп'ютерного зору.

Класичний алгоритм побудови HOG-дескриптору призводить лише до задовільної точності розпізнавання, у порівнянні з людською. Для вирішення цієї проблеми існують різні підходів, наприклад, використання інтерполяції у процесі розрахунку HOG-дескриптора [1], техніка інтегрального представлення зображення [3], впровадження згладжування тощо. Основні складнощі при даному підході полягають у значному ускладненні розрахунків та, як наслідок, пониженні швидкості розпізнавання. Однак, за останні роки почали з'являтись роботи, в яких автори розробляли підходи для покращення швидкості розрахунків HOG дескриптора.

В даній роботі реалізується модифікація алгоритму [1], що дозволяє оптимізувати процес розрахунків HOG-дескриптора, позбувшись надлишкових операцій при застосуванні інтерполяції, будуючи гістограми для кожної комірки лише один раз. Розроблена програмна реалізація даного алгоритму адаптована до можливостей використаної мови програмування Javascript та сучасних технологій обробки зображення.

Постановка задачі. Нехай маємо область розпізнавання на зображенні розміром $S_x \times S_y$, де S_x – ширина зображення у пікселях, S_y – висота зображення у пікселях. Розглянемо розбиття зображення на $C_x \times C_y$ комірок розміром $p \times p$, де C_x – кількість комірок у ширину ($C_x = S_x / p$), C_y – кількість комірок у висоту ($C_y = S_y / p$), p – розмір комірки у пікселях.

Необхідно покращити результати оптимізації розрахунку гістограми напрямлених градієнтів для розпізнавання змісту зображення, розробити програмне забезпечення, що містить імплементацію методу побудови гістограми напрямлених градієнтів із подальшою оптимізацією розрахунків [1] на базі даного зображення із виключенням надлишкових операцій та використанні інтерполяції у процесі розрахунків.

Основний матеріал. HOG-дескриптор базується на розрахунку нормалізованих локальних гістограм напрямлених градієнтів.

Маємо область розпізнавання $S_x \times S_y$ пікселів. Розрахуємо градієнт для кожного пікселя у області розпізнавання. Градієнт для x-напряму g_x, розрахуємо за рівнянням (1), градієнт для y-напряму g_y за рівнянням (2), де $f(x,y)$ являє собою значення пікселю у позиції (x,y) для зображення I.

$$g_x = \frac{\partial I}{\partial x} f(x+1,y) - f(x-1,y) \tag{1}$$

$$g_y = \frac{\partial I}{\partial y} f(x,y+1) - f(x,y-1) \tag{2}$$

Далі використаємо градієнти g_x та g_y для розрахунку магнітуди M та напрямленості θ за рівняннями (3) та (4) відповідно.

$$M(x,y) = \left(g_x{}^2 + g_y{}^2\right)^{1/2} \tag{3}$$

$$\theta(x,y) = \operatorname{ctg}\left(\frac{g_y}{g_x}\right) \tag{4}$$

Наступним кроком є накопичення «weighted votes» – звішених голосів для градієнтної магнітуди до N орієнтаціоних масивів у комірках розміром $p \times p$. Коли внутрішня дистанція масиву – θ_{dist} становить 20° на проміжку від 0° до 180°, $N = 9$, де N – кількість масивів для кожної комірки. Для уникнення згладжування, застосовується інтерполяція блоків із ціллю білінійно інтерполювати weighted votes між сусідніми масивами за напрямом та позицією. Наступним кроком є нормалізація контрасту у кожному блоці із допомогою двох схем нормалізації(L1-*norm* та L2-*norm* відповідно) представлених у рівняннях (5) та (6), де B_k – 36-ти вимірний вектор для блоку, c – кожен елемент вектору, ε – невелика константа для уникнення ділення на нуль.

$$\text{L1-}norm: \left\{ \frac{c}{\|B_k\| + \varepsilon} \right\} \tag{5}$$

$$\text{L2-}norm: \left\{ \frac{c}{\left(\|B_k\|^2 + \varepsilon^2\right)^{1/2}} \right\} \tag{6}$$

Вимір кожного блоку задається кількістю орієнтаційних масивів. Останнім кроком є сбір векторів гістограми для всіх пересічних блоків у області розпізнавання. Коли $S_x = 48$, $S_y = 48$, $p = 6$, $c = 2$, $L = 6$, де L – крок блоку та $N = 9$, то загальна кількість блоків у області розпізнавання становить 49.

Для підвищення точності розпізнавання, при розрахунку HOG застосовується техніка інтерполяції з ціллю уникнення згладжування та покращення точності розпізнавання. Приклад білінійної та трилінійної інтерполяції продемонстровано на рис. 1, де $C_0 \sim C_3$ – сусідні комірки. Коли $N = 9$ на від 0°

до 180° та напрям градієнту для пікселю у $C_0 = 37°$, два сусідні масиви зазначені як 1 та 2 відповідно. Оскільки у лінійній інтерполяції враховується лише напрям пікселю, weighted votes розраховуються шляхом помноження магнітуди – M, коефіцієнт – G та зважження напряму – W_θ. Потім weighted votes розбиваються на два масиви (масив 1 та 2 лише для C_0). З іншого боку, напрям та позиція пікселя у блоку розглядається у трилінійній інтерполяції, тому у цьому випадку weighted votes розраховуються множенням магнітуди – M, коефіцієнту – G, зваженням напряму – W_θ та зважень для позиції пікселю (W_x та W_y відповідно), а потім розбиваються на вісім масивів (масив 1 та 2 для $C_0 \sim C_3$) (див. рис. 1.)

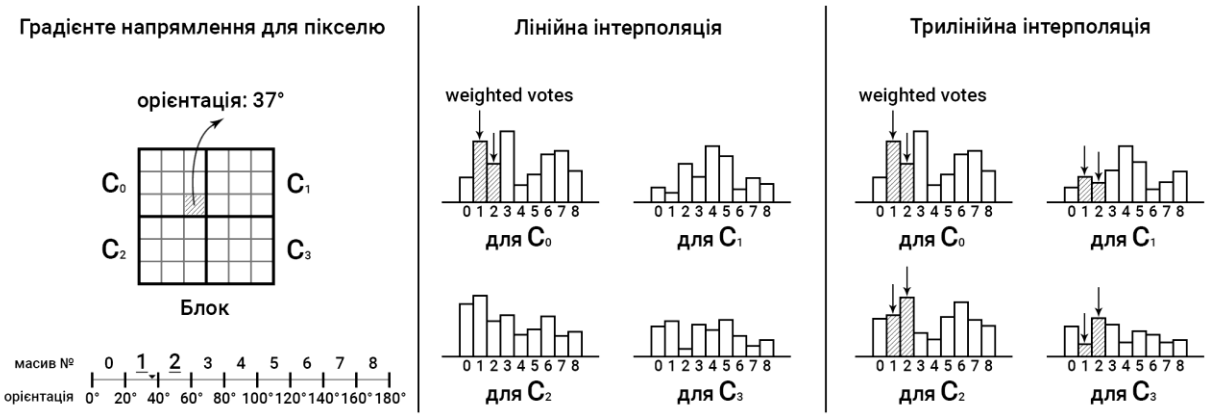

Рис. 1 Інтерполяція блоку

Незважаючи на факт, що трилінійна інтерполяція значно покращує точність, швидкість розпізнавання зменшується через високу складність розрахунків. Оскільки HOG-дескриптор розраховується із врахуванням всіх пересічних блоків у області розпізнавання, кількість розрахунків підвищується й надалі. Таблиця 1 демонструє кількість необхідних множень на області розпізнавання, враховуючи інтерполяцію та L2-*norm* нормалізацію, де B_x – кількість блоків у ширину, B_y – кількість блоків у висоту. Вочевидь, кількість необхідних множень для трилінійної інтерполяції набагато більше ніж для лінійної. Тому для підвищення швидкості розпізнавання більшість дослідників як правило ігнорують трилінійну інтерполяцію та часто застосовують лінійну інтерполяцію.

Таблиця 1

Інтерполяція	Кількість множень на області розпізнавання
Лінійна	$\left\{\left(4\times p^2\times c^2\right)+\left(c^2\times N\right)+1\right\}\times B_x\times B_y$
Трилінійна	$\left[\begin{array}{l}\left\{32\times p^2\times\left(c-1\right)^2\right\}+\left\{32\times p^2\times\left(c-1\right)\right\}\\+\left(8\times p^2\right)+\left(c^2\times N\right)+1\end{array}\right]\times B_x\times B_y$

Алгоритм оптимізації швидкості розрахунків. Застосуємо алгоритм, що виконує задачу оптимізації, дозволяє повністю уникнути надлишкових операцій, уникаючи розрахунку однієї комірки двічі, що значно підвищує швидкість розпізнавання, не втрачаючи точність розпізнавання.

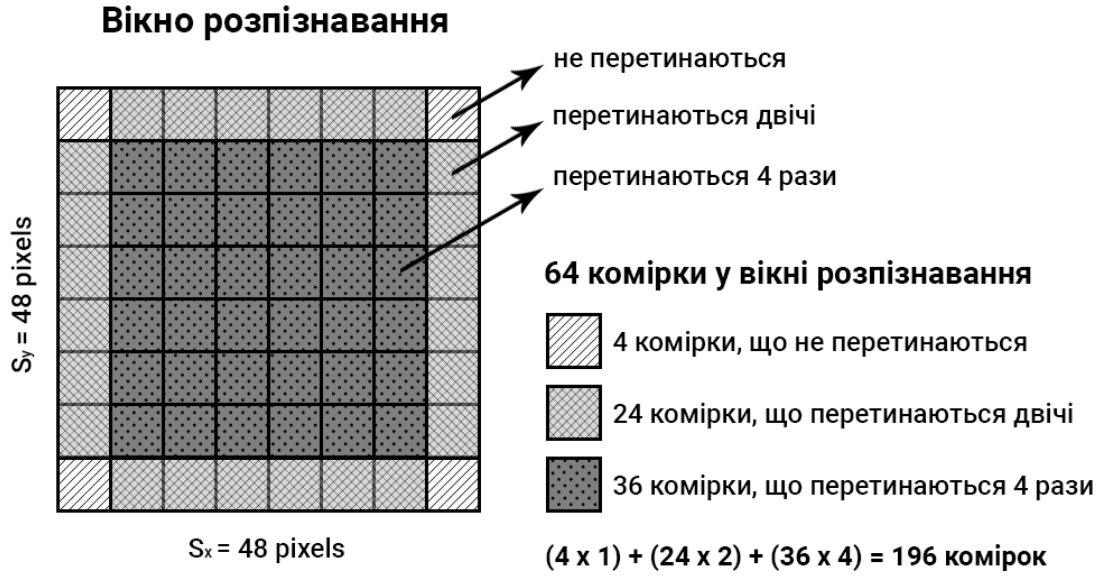

Рис. 2. Кількість необхідних комірок у розрахунку HOG-дескриптора

Надлишкові операції присутні у оригінальному алгоритмі через операції над пересічними частинами блоків. Коли $S_x = 48$, $S_y = 48$, $c = 2$ та $L = p = 6$, наприклад, 24 комірки перетинаються двічі та 36 комірок перетинаються чотири рази, як вказано на рис. 2. Хоча кількість комірок у області розпізнавання становить 64, кількість комірок, необхідних для розрахунку HOG - дескриптора становить 196 блоки.

Кожен блок у області розпізнавання розбивається на чотири умовні регіони – $SB_1 \sim SB_4$, як вказано на рис. 3. Оскільки інтервал кожного пересічного блоку становить одну комірку у кожному напрямі, більшість комірок належать до чотирьох заздалегідь визначених типів.

1. [масив напряму] $\leftarrow M \times G_1 \times W_x \times W_y \times W_\theta$
2. [масив напряму] $\leftarrow M \times G_2 \times W_x \times W_y \times W_\theta$
3. [масив напряму] $\leftarrow M \times G_3 \times W_x \times W_y \times W_\theta$
4. [масив напряму] $\leftarrow M \times G_2 \times W_x \times W_y \times W_\theta$

Кожній комірці присвоюється відповідний тип відповідно до кількості пересічних блоків, що містять дану комірку. Наприклад, якщо лише один блок містить комірку, вона містить один тип, при перетинанні двох блоків – 2 типи, при перетинанні чотирьох блоків – усі чотири типи. Як зазначено у відповідних формулах розрахунків, лише коефіцієнт Гауса відрізняється відповідно до типу комірки($G_1 \sim G_4$ являють собою коефіцієнти Гауса у регіонах

$SB_1 \sim SB_4$). Отже рівняння трилінійної інтерполяції модифікується із ціллю розділу загальних операцій для кожного типу комірки.

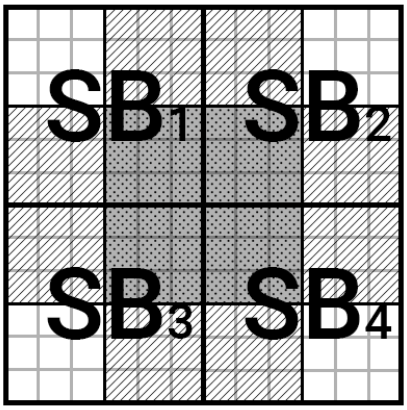

Рис. 3 Розбиття блоку на регіони

Розрахуємо трилінійну інтерполяцію для кожного з чотирьох типів регіонів комірки – $SC_1 \sim SC_4$, де α та β – коефіцієнти інтерполяції, k – індекс групи масивів комірки. Тип 1:

$$SC_1 : \left[Nk + \alpha\right] \leftarrow M \times A \times G_1 \times \beta$$

$$SC_2 : \begin{cases} \left[Nk + \alpha\right] \leftarrow M \times A \times G_1 \times \beta \\ \left[N\left(k+1\right) + \alpha\right] \leftarrow M \times C \times G_1 \times \beta \end{cases}$$

$$SC_3 : \begin{cases} \left[Nk + \alpha\right] \leftarrow M \times A \times G_1 \times \beta \\ \left[N\left(k+2\right) + \alpha\right] \leftarrow M \times B \times G_1 \times \beta \end{cases}$$

$$SC_4 : \begin{cases} \left[Nk + \alpha\right] \leftarrow M \times A \times G_1 \times \beta \\ \left[N\left(k+1\right) + \alpha\right] \leftarrow M \times C \times G_1 \times \beta \\ \left[N\left(k+2\right) + \alpha\right] \leftarrow M \times B \times G_1 \times \beta \\ \left[N\left(k+3\right) + \alpha\right] \leftarrow M \times D \times G_1 \times \beta \end{cases}$$

Тип 2:

$$SC_1 : \begin{cases} \left[N\left(k-4\right) + \alpha\right] \leftarrow M \times C \times G_2 \times \beta \\ \left[N\left(k-3\right) + \alpha\right] \leftarrow M \times A \times G_2 \times \beta \end{cases}$$

$$SC_2 : \left[N\left(k-3\right) + \alpha\right] \leftarrow M \times A \times G_2 \times \beta$$

$$SC_3 : \begin{cases} \left[N\left(k-4\right) + \alpha\right] \leftarrow M \times C \times G_2 \times \beta \\ \left[N\left(k-3\right) + \alpha\right] \leftarrow M \times A \times G_2 \times \beta \\ \left[N\left(k-2\right) + \alpha\right] \leftarrow M \times D \times G_2 \times \beta \\ \left[N\left(k-1\right) + \alpha\right] \leftarrow M \times B \times G_2 \times \beta \end{cases}$$

$$SC_4:\begin{cases}\left[N\left(k-3\right)+\alpha\right]\leftarrow M\times A\times G_2\times\beta\\ \left[N\left(k-1\right)+\alpha\right]\leftarrow M\times B\times G_2\times\beta\end{cases}$$

Тип 3:

$$SC_1:\begin{cases}\left[N\left(k-28\right)+\alpha\right]\leftarrow M\times B\times G_3\times\beta\\ \left[N\left(k-26\right)+\alpha\right]\leftarrow M\times A\times G_3\times\beta\end{cases}$$

$$SC_3:\begin{cases}\left[N\left(k-28\right)+\alpha\right]\leftarrow M\times B\times G_3\times\beta\\ \left[N\left(k-27\right)+\alpha\right]\leftarrow M\times D\times G_3\times\beta\\ \left[N\left(k-26\right)+\alpha\right]\leftarrow M\times A\times G_3\times\beta\\ \left[N\left(k-25\right)+\alpha\right]\leftarrow M\times C\times G_3\times\beta\end{cases}$$

$$SC_3:\left[N\left(k-26\right)+\alpha\right]\leftarrow M\times A\times G_3\times\beta$$

$$SC_4:\begin{cases}\left[N\left(k-26\right)+\alpha\right]\leftarrow M\times A\times G_3\times\beta\\ \left[N\left(k-25\right)+\alpha\right]\leftarrow M\times C\times G_3\times\beta\end{cases}$$

Тип 4:

$$SC_1:\begin{cases}\left[N\left(k-32\right)+\alpha\right]\leftarrow M\times D\times G_4\times\beta\\ \left[N\left(k-31\right)+\alpha\right]\leftarrow M\times B\times G_4\times\beta\\ \left[N\left(k-30\right)+\alpha\right]\leftarrow M\times C\times G_4\times\beta\\ \left[N\left(k-29\right)+\alpha\right]\leftarrow M\times A\times G_4\times\beta\end{cases}$$

$$SC_2:\begin{cases}\left[N\left(k-31\right)+\alpha\right]\leftarrow M\times B\times G_4\times\beta\\ \left[N\left(k-29\right)+\alpha\right]\leftarrow M\times A\times G_4\times\beta\end{cases}$$

$$SC_3:\begin{cases}\left[N\left(k-30\right)+\alpha\right]\leftarrow M\times C\times G_4\times\beta\\ \left[N\left(k-29\right)+\alpha\right]\leftarrow M\times A\times G_4\times\beta\end{cases}$$

$$SC_4:\left[N\left(k-29\right)+\alpha\right]\leftarrow M\times A\times G_4\times\beta$$

У даних рівняннях $A\sim D$ являють собою зважения для позиції пікселю – $W_x\times W_y$), зазначених у рівняннях (7), (8), (9) та (10) відповідно, де $\left(x,y\right)$ – позиція пікселю.

$$\begin{cases}x'=\left(x+0.5\right)/\ p-0.5\\ y'=(y+0.5)/\ p-0.5\end{cases}\Rightarrow A\leftarrow X_H\times Y_H \qquad (7)$$

$$\begin{cases}x''=\ floor\left(x'\right)\\ y''=\ floor(y')\end{cases}\Rightarrow B\leftarrow X_H\times\left(1-Y_H\right) \qquad (8)$$

$$\begin{cases} x_1 = x' - x'' \\ y_1 = y' - y'' \\ x_2 = 1 - x_1 \\ y_2 = 1 - y_1 \end{cases} \Rightarrow C \leftarrow (1 - X_H) \times Y_H \qquad (9)$$

$$\begin{cases} X_H = \max(x_1, x_2) \\ Y_H = \max(y_1, y_2) \end{cases} \Rightarrow D \leftarrow (1 - X_H) \times (1 - Y_H) \qquad (10)$$

Слід зауважити, що $G_1 \sim G_4$ та $A \sim D$ розраховані заздалегідь, оскільки вони вже визначені за p та c. Модифіковані рівняння трилінійної інтерполяції використовуються у подальшому розрахунку HOG дескриптора за запропонованим алгоритмом.

Програмна реалізація алгоритму. Швидкість розрахунків при програмній реалізації залежить не лише від реалізованого алгоритму, але й від оптимального використання операцій, наданих мовою програмування Javascript.

Оскільки алгоритм оптимізації розрахунків базується на операціях на рівні комірки, необхідно розбити вхідне зображення на масив комірок, відповідно до вхідних параметрів ($p \times p$ – розмір комірки, $S_x \times S_y$ – розмір області розпізнавання). Виконаємо розрахунок, створивши функцію *getImageCells* схематично представлений на рис. 4. Слід зауважити, що розбиття коду на функції (модульність) – є невід'ємним стандартом мови програмування Javascript.

```
getImageCells(image, p) {
  const cells = new Array((image.length / p) * (image[0].length / p));
  let cellIndex = 0;

  for (let i = 0; i < image.length; i++) {
    for (let j = 0; j < image[0].length; j++) {
      clusterizeImage(cells, image[i][j], p);
      calculateGaussian(image[i][j]);
      calculateAD(image[i][j]);
    }
  }

  return cells;
}
```

Рис. 4 Функція розбиття зображення на комірки

Оскільки проходження по двомірному масиві – досить важка операція, зменшимо кількість надлишкових операцій втричі, включивши до проходження по пікселям зображення розрахунок коефіцієнтів Гауса (*calculateGaussian*) та коефіцієнтів $A \sim D$ (*calculateAD*).

Маючи масив комірок та розраховані коефіцієнти для кожного пікселя, наступним кроком є розрахунок HOG дескриптора, що потребує повторного проходження по кожному пікселю, але тепер на рівні комірки. Виконаємо ро-

зрахунок, створивши функцію *getHOGfeature*, що містить у собі необхідну логіку для подальшої реалізації методу.

```
getHOGfeature(cells) {
  let hog = null;

  for (let i = 0; i < cells.length; i++) {
    for (let x = 0; x < cells[i].length; x++) {
      for (let y = 0; y < cells[i][x].length; y++) {
        const { n, n1, n2, na, nb } = calculateOrientationWeights(cells[i][x][y]);
        const { alpha, beta } = calculateCoefficients(n, n1, n2, na, nb);

        applyInterpolation(alpha, beta);
        hog = calculateHOG(cells[i][x][y], alpha, beta);
      }
    }
  }

  return hog;
}
```

Рис. 5 Функція розрахунку HOG дескриптора

Розрахунок HOG дескриптора включає в себе розрахунок зважених голосів для трилінійної інтерполяції (*calculateOrientationWeights*), коефіцієнти A та B для трилінійної інтерполяції (*calculateCoefficients*), трилінійної інтерполяції (*applyInterpolation*) та безпосередньо гістограми напрямлених градієнтів (*calculateHOG*).

Функція *calculateHOG* містить у собі розрахунок відповідно до одного з чотирьох запропонованих типів комірки. Наприклад, коли піксель знаходиться у позиції $(0,0)$ у 0-ій комірці, роздивляється тільки тип 1, оскільки 0-ва комірка належить лише одному блоку, тому не причепна до перетинання, а коли піксель знаходиться у позиції $(7,0)$ у 1-ій комірці, належить до типу 2 у 0-му блоці та до типу 1 у 1-му блоці.

Незважаючи на ускладнення реалізації через більш складну логіку алгоритму оптимізації, даний метод надає можливість позбавитися від значної кількості надлишкових операцій, що прискорює розрахунок гістограми напрямлених градієнтів.

Таким чином отримуємо HOG дескриптор, що достатньо швидкий у розрахуванні та завдяки наявності інтерполяції при розрахунку, надає кращу точність при розпізнавання, ніж класичний HOG дескриптор.

Висновки. Для програмної реалізації, було обрано та програмно реалізовано метод оптимізації, представлений у роботі [1]. Було покращено результати оптимізації розрахунку HOG-дескриптора для розпізнавання змісту зображення. Для того, щоб повністю позбавитись надлишкових операцій, HOG дескриптор розраховувався на рівні операцій над коміркою. Такий підхід до-

зволив розгляну кожну комірку лише один раз, що значно покращило швидкість розрахунку та, як наслідок, швидкість розпізнавання образів.

Тестування програмної реалізації на вище зазначеному наборі вказало, що даний підхід не тільки надає високу точність розпізнавання, але й покращує швидкість проведення розрахунків, побудови гістограми напрямлених градієнтів.

Програмна реалізація виконана на мові програмування Javascript та відповідає стандартам відповідного програмного забезпечення.

Бібліографічні посилання

1. **Soojin, Kim.** Fast Calculation of Histogram of Oriented Gradient Feature by Removing Redundancy in Overlapping Block [Text] / Soojin Kim, Kyeongsoon Cho // Journal of Information Science and Engineering. – 2014 – Volume 30. – С.1719–1731.
2. **Navneet, Dalal.** Histograms of Oriented Gradients for Human Detection [Text] / Navneet Dalal, Bill Triggs // 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) – 2005. – Volume 1. – С. 886–893.
3. **Южаков, Г.Б.** Алгоритм быстрого построения дескрипторов изображения, основанных на технике гистограмм ориентированных градиентов [Текст] / Г.Б Южаков // Труды МФТИ. – 2013. – №3. – С. 84–91.

Надійшла до редколегії 30.09.2019.