Министерство образования и науки Российской Федерации
Национальный исследовательский технологический университет «МИСИС»
Институт компьютерных наук
Кафедра Инженерной Кибернетики

Курсовая работа

по дисциплине «Объектно-ориентированное программирование»
на тему «Библиотека для извлечения признаков HOG»

Выполнил:
студент гр. БПМ-22-1
Кадомцев А.Р.

Проверил:
доцент кафедры КИК, к.т.н.
Полевой Д.В.

Москва, 2023

Оглавление

1. Описание задачи

Данная курсовая работа посвящена разработке библиотеки для работы с признаками Histogram of oriented gradients (Гистограмма ориентированных градиентов) (далее HOG) изображения, а также тестового приложения, демонстрирующего работу с ней. Задачей является создание функционального инструмента (библиотеки) для извлечения признаков HOG с изображения, который может быть использован в различных приложениях, связанных с компьютерным зрением и обработкой изображений.

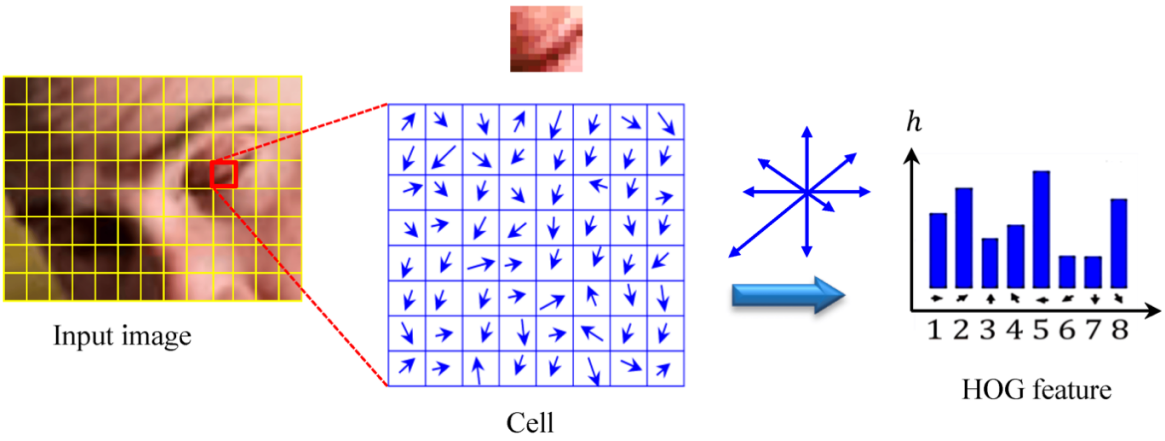

Рисунок 1: Гистограмма градиентов ячейки изображения [1]

В основные функции библиотеки входят:

- Извлечение признаков HOG с изображения
- Визуализация признаков HOG
- Визуализация процесса извлечения признаков

В основные функции тестового приложения входят:

- Извлечение признаков с изображения
- Визуализация извлеченных признаков
- Визуализация процесса по извлечению признаков
- Возможность изменить параметры алгоритма

***Примечание**: тестовое приложение является примером использования библиотеки и может отражать не весь ее функционал.*

Требования к технической реализации:

- Язык программирования C++
- Сборка с использованием CMake
- Кроссплатформенная сборка
- Автоматическая генерация Doxygen документации

2. Математическая модель алгоритма

Математическая модель алгоритма извлечения признаков HOG включает следующие шаги:

1. Предварительная обработка изображения:
   a. Вычисление градиентов изображения, чтобы получить информацию о направлении и силе изменения яркости пикселей [2].

   Горизонтальный градиент (G_x) : $G_x = [-1;\ 0;\ 1] * I$
   Вертикальный градиент$\left(G_y\right)$: $G_y = [-1; 0; 1] * I$
   Направление градиента(θ) : $\theta(i,j) = arctan\left(G_y(i,j)/G_x(i,j)\right)$
   Величина градиента(M) : $M(i,j) = sqrt\left(G_x(i,j)^2 + G_y(i,j)^2\right)$

2. Разделение изображения на небольшие ячейки:
   a. Изображение разделяется на множество неперекрывающихся ячеек фиксированного размера.
   (Обычно используется размер ячейки 8x8 пикселей).

3. Вычисление гистограмм направленных градиентов в каждой ячейке:
   a. Для каждой ячейки вычисляются направления градиентов и их величины.
   b. Интервал значений направлений градиентов разбивается на несколько корзин (обычно 9), и градиенты размещаются в соответствующих корзинах [3].
   c. Для каждой корзины вычисляется сумма величин градиентов, что создает гистограмму направленных градиентов для ячейки.

4. Создание блоков из нескольких ячеек:
   a. Несколько ячеек объединяются в блоки (обычно 2x2 ячейки) с целью учесть локальную структуру объекта.
   b. Блоки могут перекрываться.

5. Нормализация блоков:
   a. В каждом блоке выполняется нормализация гистограммы [4]

   $f = \frac{\mathrm{v}}{\sqrt{|\mathrm{v}|_2^2 + \mathrm{e}^2}}$ с ограничением сверху в 0.5 $(L2 - hys)$

   Где v – вектор содержащий все гистограммы блока

$|v|_k$ — его $k-$ норма при $k = 1{,}2$ и e некая малая константа

6. Получение признакового вектора:
    a. Гистограммы блоков объединяются в один признаковый вектор.
    b. Этот вектор представляет собой финальное представление изображения.

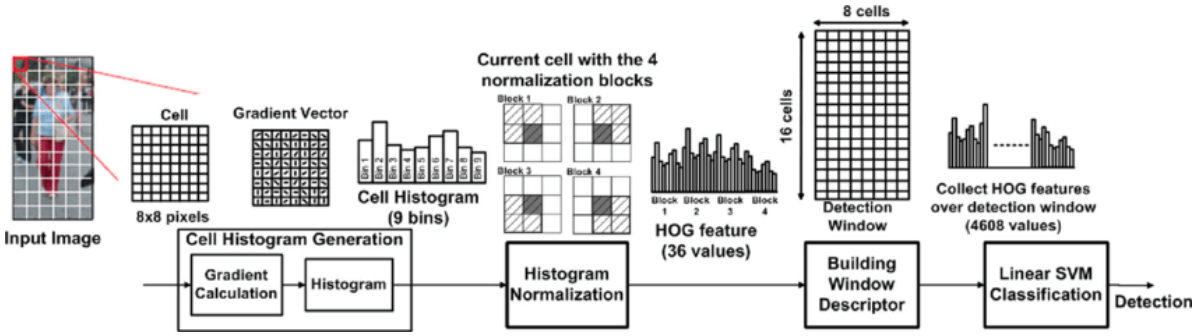

Рисунок 2: Процесс извлечения признаков HOG [5]

3. Сборка и установка проекта

Требования к пользователю:

1. Менеджер пакетов Vcpkg
2. Библиотека OpenCV
3. Doxygen (Для генерации документации)

Инструкция

1. Склонируйте проект по ссылке себе на компьютер https://github.com/Aven1r/cpp_2nd_semester
2. В терминале, зайдите в папку /kadomtsev_a_r и введите следующие команды
    a. Для linux/macOS

```
cmake -DCMAKE_TOOLCHAIN_FILE=[path] -
DCMAKE_INSTALL_PREFIX=[path2] -B ./build
cd ./build
cmake --build .
cmake –-install .
```

    b. Для Windows

```
cmake -DCMAKE_TOOLCHAIN_FILE=[path] -
DCMAKE_INSTALL_PREFIX=[path2] -B ./build
```

```
cd ./build
cmake --build . --target install
```

Где path – путь до менеджера библиотек, а path2 – путь, куда будет установлена библиотека/программа.

Примечание: *по умолчанию проект попытается сгенерировать документацию Doxygen и скомпилировать тестовое приложение. Чтобы исключить их из сборки, вы можете использовать следующие флаги в первой команде CMake:*

```
-DDOXYGEN=OFF -DSAMPLE=OFF
```

Внимание*: если указан параметр сборки* `CMAKE_INSTALL_PREFIX`*, то другие проекты не смогут воспользоваться библиотекой hoglibrary, пока указанный путь не будет добавлен в системные переменные окружения (т.н. PATH).*

Если же параметр сборки `CMAKE_INSTALL_PREFIX` *не был указан, но опция* `SAMPLE` *была активирована, то из-за ограничений системы, пользователь потеряет доступ к функционалу с изменением параметров в приложении.*

4. Пользовательское описание

4.1. Библиотека

В своем коде, пользователь может использовать[1] функционал библиотеки. Для этого нужно добавить следующую строку в свой CMakeLists проекта:

```
find_package(hoglibrary REQUIRED)
target_link_libraries(target_name PRIVATE hoglibrary)
```

И добавить следующие заголовки в свой c++ файл:

```
#include <hogdescriptor/hogdescriptor.hpp>
//Для работы с дескриптором
#include <texvisualization/texvisualization.hpp>
//Для сохранения .tex файлов
```

[1] При условии, что пользователь уже добавил в окружение проекта библиотеку OpenCV, а также установил папку с проектом в системное окружение.

Библиотека состоит из 2 классов: HOGDescriptor и texHOG.

После создания объекта класса HOGDescriptor, пользователь получает доступ к следующим методам библиотеки hoglibrary:

- **visualizeHOG -** *Метод для визуализации гистограмм ячеек* ***HOGDescriptor*** *в виде стрелок внутри каждой ячейки на изображении*
- **HOGgrid -** *Метод для отображения сетки ячеек на изображении*

- **computeHOG -** *Метод для вычисления гистограмм HOG.*
- **getHOGFeatureVector -** *Метод для получения вектора гистограммы HOG.*
- **getCellHistogram** - *Получение гистограммы ячейки*
- **getBlockHistogram** - *Получение гистограммы блока*
- **saveVectorData** - *Сохранение вектора HOG в файл*

Примечание: *за подробной информацией о возвращаемых типах и аргументах методов, обратитесь к документации и/или технической части отчета.*

После создания объекта класса texHOG, пользователь получает доступ к следующим методам:

- **cellHistogramPlot -** *Метод для создания .tex файла с графиком гистограммы ячейки*
- **blockHistogramPlot** - *Метод для создания файла .tex с гистограммами ячеек в заданном блоке*

Примечание: *за подробной информацией о возвращаемых типах и аргументах методов, обратитесь к документации и/или технической части отчета.*

4.2. Консольное приложение

После сборки и установки, пользователь может воспользоваться консольным приложением `hogexe` и взаимодействовать с ним через параметры командной строки.

В случае отсутствия параметров приложения или наличия только одного параметра `--help` или `-h`, выводится информация о всех возможных вариантах использования `hogexe`.

Ниже приведен вывод программы при вводе в консоль `./hogexe -h`:

```
Использование программы:
./hogexe -test [1-3]: Демонстрация работы алгоритма на выбранном примере (1, 2 или 3)
./hogexe -settings: показать настройки программы и алгоритма
./hogexe -p <path to image> : Демонстрация работы алгоритма на вашем изображении
```

5. Техническое описание

5.1. Библиотека

Класс HOGDescriptor:

Открытые члены:

- **HOGDescriptor** ()
  *Конструктор для инициализации объекта класса **HOGDescriptor** с параметрами по умолчанию*
- **HOGDescriptor** (const size_t blockSize, const size_t cellSize, const size_t stride, const size_t binNumber, const size_t gradType)
  *Конструктор для класса **HOGDescriptor** со всеми параметрами*
- **HOGDescriptor** (const size_t blockSize, const size_t cellSize)
  *Конструктор для создания нового объекта **HOGDescriptor**.*
- **~HOGDescriptor** ()
  *Деструктор для класса **HOGDescriptor**.*

- void **visualizeHOG** (float scale, bool imposed)
  *Метод для визуализации гистограмм ячеек **HOGDescriptor** в виде стрелок внутри каждой ячейки на изображении*
- void **HOGgrid** (cv::Mat &image, float thickness, int cellSize)
  Метод для отображения сетки ячеек на изображении

- void **computeHOG** (cv::Mat &image)
  Метод для вычисления гистограмм HOG.
- std::vector< float > **getHOGFeatureVector** ()
  Метод для получения вектора гистограммы HOG.
- std::vector< float > **getCellHistogram** (int y, int x)
  Получение гистограммы ячейки
- std::vector< std::vector< float > > **getBlockHistogram** (int y, int x)
  Получение гистограммы блока

- void **saveVectorData** (const std::string &executablePath, const std::string &vectorName)
  Сохранение вектора HOG в файл

Статистические открытые члены

- static const size_t **GRADIENT_SIGNED** = 360
  Разброс градиента на 360 градусов
- static const size_t **GRADIENT_UNSIGNED** = 180
  Разброс градиента на 180 градусов

Закрытые члены

- void **computeGradientFeatures** (cv::Mat &image)
  Функция для вычисления амплитуды и ориентации градиента каждого пикселя
- std::vector< std::vector< std::vector< float > > > **computeCellHistograms** (cv::Mat magnitude, cv::Mat orientation, std::vector< std::vector< std::vector< float > > > &cell_histograms)
  Вычисление гистограмм HOG для каждой ячейки изображения.

- std::vector< float > **cellHistogram** (const cv::Mat &cellMagnitude, const cv::Mat &cellOrientation)
  Метод для вычисления гистограммы для данной ячейки
- void **normalizeBlockHistogram** (std::vector< float > &block_histogram)
  Функция для нормализации значений из гистограмм HOG для объедененных ячеек из блока

- const std::vector< float > **calculateHOGVector** (const std::vector< std::vector< std::vector< float > > > &cell_histograms)
  Метод для вычисления вектора гистограммы HOG.

Закрытые данные

- int **blockSize_**
  Размер блока скользящего окна в пикселях
- int **cellSize_**
  Размер ячейки в пикселях
- int **binNumber_**
  Количество корзин в гистограмме каждой ячейки
- int **binWidth_**
  Ширина корзин в гистограмме каждой ячейки
- int **stride_**
  Шаг скользящего окна в пикселях
- int **gradType_**
  Тип вычисления градиента (беззнаковый или со знаком)
- bool **hogFlag_** = false
  Флаг, указывающий, был ли вычислен вектор гистограммы HOG.
- cv::Mat **imageMagnitude_**
  Амплитуда градиента изображения
- cv::Mat **imageOrientation_**
  Ориентация градиента изображения
- std::vector< std::vector< std::vector< float > > > **cellHistograms_**
  Вектор гистограмм ячеек
- std::vector< float > **hogFeatureVector_**
  Финальный вектор гистограммы HOG.

Класс texHOG:

Открытые члены

- **texHOG** ()=default
  Конструктор класса

- void **cellHistogramPlot** (std::vector< float > cellHistogram, int binWidth, const std::string &executablePath, const std::string& plotName)
  Метод для создания .tex файла с графиком гистограммы ячейки
- void **blockHistogramPlot** (std::vector< std::vector< float > > blockHistogram, int binWidth, const std::string &executablePath, const std::string& plotName)
  Метод для создания файла .tex с гистограммами ячеек в заданном блоке

5.2. Консольное приложение

Результат выполнения команды `./hogexe -h`:

```
Параметры успешно загружены из файла hogconfig!
Использование программы:
./hogexe -test [1-3]: Демонстрация работы алгоритма на выбранном примере (1, 2 или 3)
./hogexe -settings: показать настройки программы и алгоритма
./hogexe -p <path to image> : Демонстрация работы алгоритма на вашем изображении
```

Результат выполнения команды `./hogexe -test 1`:

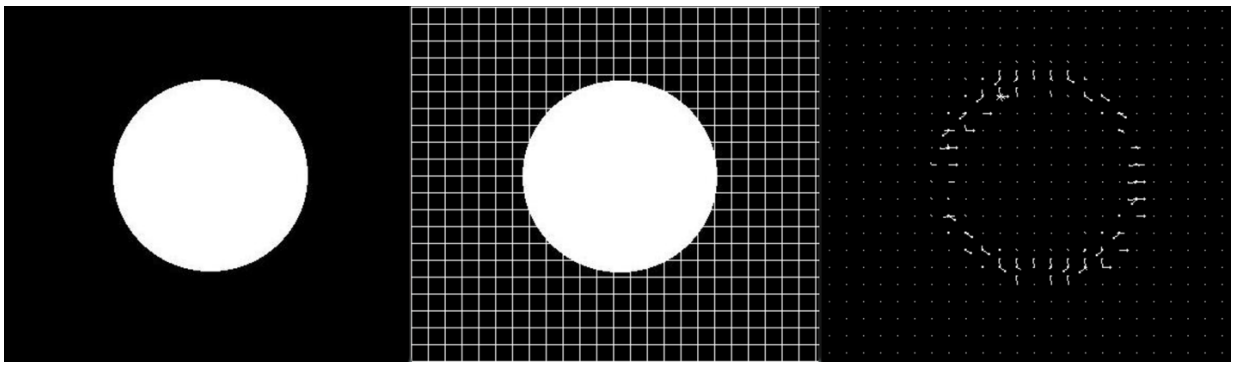

Рисунок 3: Результат выполнения команды ./hogexe -test 1
(Слева направо: 1) Исходное изображение 2) Изображение с размеченной сеткой ячеек 3) Визуализация вектора HOG на изображении)

Результат выполнения команды `./hogexe -settings`:

```
Параметры успешно загружены из файла hogconfig!
--------------------Текущие настройки----------------------
Block size: 16
Cell size: 8
Stride: 8
Bin number: 9
Gradient type: 180
Bin width: 20
-----Прочее-----
Путь для сохранения данных: /Users/avenir/vscode/
Разрешение на сохранение итогового HOG вектора: 0
Разрешение на сохранение .tex файлов с графиками: 0
-------------------------------------------------------
Чтобы сменить параметры алгоритма введите 1
Чтобы изменить путь для сохранения данных введите 2
Чтобы изменить разрешение на сохранение вектора введите 3
Чтобы изменить разрешение на сохранение .tex файлов введите
4
Чтобы выйти из меню введите любое другое значение
```

При вводе 1 пользователю предоставляется возможность изменить параметры алгоритма:

```
...
1
Введите новое значение Block Size
16
Введите новое значение Cell Size
8
Введите новое значение Gradient Type (360 or 180)
180
Настройки сохранены
Изменения сохранены
```

При вводе 2 пользователь может изменить путь, где будут сохраняться файлы, если это возможно:

```
...
2
Введите новый путь для сохранения данных
/Users/avenir/vscode/
Настройки сохранены
Изменения сохранены
```

Ввод `3` или `4` снимает/дает разрешение на сохранение файлов

Результат выполнения команды `./hogexe -test 2` с включенными параметрами для сохранения файлов:

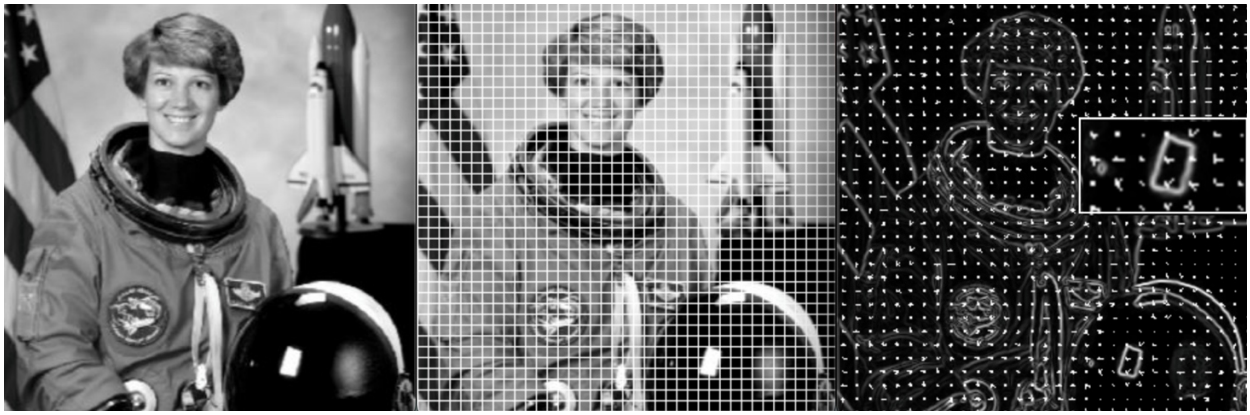

Рисунок 4 Результат выполнения команды ./hogexe -test 2
(Слева направо: 1) Исходное изображение 2) Изображение с размеченной сеткой ячеек 3) Визуализация вектора HOG на изображении)

Сгенерированный test2_cellhist.tex

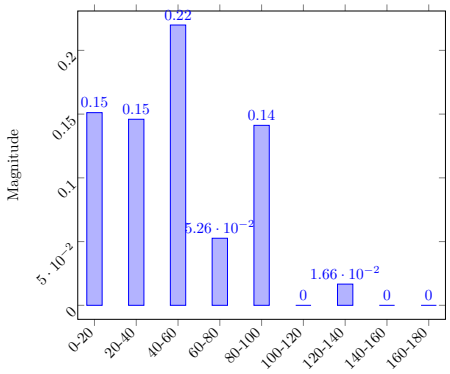

Рисунок 5 Гистограмма ориентированных градиентов ячейки (7,9)

Сгенерированный test2_blockhist.tex

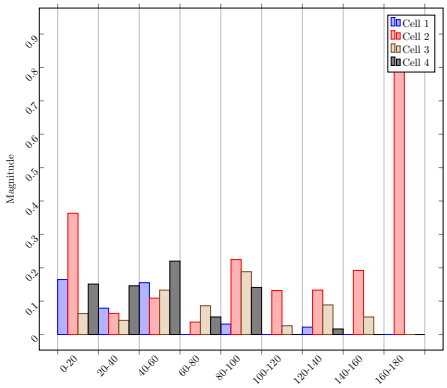

Рисунок 6 Гистограмма ориентированных градиентов для блока с первой ячейкой (6,8)

Внимание: *при попытке проверить программу на пользовательских изображениях, стоит учесть, что на вход принимаются только изображения в оттенках серого. При попытке загрузить изображение другого типа, программа выдаст предупреждение и прекратит свою работу.*

Результат выполнения команды `./hogexe -p /path/to/image` на примере пути до цветного изображения:

```
Параметры успешно загружены из файла hogconfig!
libc++abi: terminating due to uncaught exception of type
std::runtime_error: The image is not grayscale!
```

Результат выполнения команды `./hogexe -p /path/to/image` на примере пути до 3 тестового изображения и параметрами `blockSize 50, cellSize 25, gradType 180`

Рисунок 7 Результат выполнения команды ./hogexe -p /Users/avenir/vscode/hoglibrary/bin/images/test3.jpg
(Слева направо: 1) Исходное изображение 3) Изображение с размеченной сеткой ячеек 3) Визуализация вектора HOG на изображении)

6. Документация

Внутри корневой папки проекта находится папка hoglib_doc, в которой сгенерирована документация проекта программой Doxygen в HTML, latex и RTF форматах. Открыв файл ~/hoglib_doc/html/index.html в браузере, вы сможете ознакомится со всеми открытыми пользователю классами и их составляющими.

7. Список использованной литературы

1. A Framework for Instantaneous Driver Drowsiness Detection Based on Improved HOG Features and Naïve Bayesian Classification // MDPI [Электронный ресурс]. URL: https://www.mdpi.com/2076-3425/11/2/240 (дата обращения: 15.06.2023).
2. Histogram of Oriented Gradients explained using OpenCV // LearnOpenCV [Электронный ресурс]. URL: https://learnopencv.com/histogram-of-oriented-gradients/ (дата обращения: 24.05.2023)
3. Dalal, Navneet; Triggs, Bill. Histograms of Oriented Gradients for Human Detection // 2005.
4. Histogram of oriented gradients // Wikipedia [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients (дата обращения: 02.06.2023).
5. Suleiman, Amr; Sze, Vivienn. Energy-Efficient Hardware Implementation of HOG-Based Object Detection at 1080HD60 fps with Multi-Scale Support // 2015.