Нижегородский государственный университет им. Н.И. Лобачевского

Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP

Лабораторная работа
Оптимизация и распараллеливание вычислений в задаче детектирования объектов на изображениях с использованием алгоритма Latent SVM

При поддержке компании Intel

Козинов Е.А., Кустикова В.Д., Половинкин А.Н.,
кафедра математического обеспечения ЭВМ

Содержание

- Цели и задачи работы
- Содержательная постановка задачи детектирования объектов
- Математическая постановка задачи детектирования объектов с n частями
- Основные этапы решения задачи детектирования с использованием алгоритма Latent SVM
- Обзор базовой программной реализации Latent SVM
- Этапы оптимизации и распараллеливания исходной программной реализации
- Оценка качества поиска объектов разных классов с использованием финальной параллельной реализации

Цели работы

- Для большинства практических приложений процедура поиска объектов на изображении должна выполняться в реальном времени.

- Основная цель данной работы – максимально уменьшить среднее время поиска объектов разных классов с использованием метода Latent SVM за счет компиляторной и алгоритмической оптимизации, а также за счет последующего распараллеливания последовательной реализации на системы с общем памятью.

Задачи работы (1)

- Изучение схемы работы алгоритма Latent SVM.
- Изучение структуры модулей и функций программной реализации Latent SVM.
- Определение участков кода, на исполнение которых тратится наибольшее время, с помощью инструментов Intel Parallel Studio XE.
- Компиляторная и алгоритмическая оптимизация наиболее трудоемких участков исходного кода.

Задачи работы (2)

- Разработка и реализация различных схем распараллеливания с использованием технологии OpenMP и библиотеки Intel® Threading Building Blocks.
- Анализ масштабируемости разработанных параллельных реализаций.
- Оценка качества детектирования объектов при использовании наиболее эффективной параллельной реализации.

Тестовая инфраструктура

Операционная система	Microsoft Windows 7
Среда разработки	Microsoft Visual Studio 2010
Библиотека TBB	Intel® Threading Building Blocks (в составе Intel® Parallel Studio XE 2013)
Профилировщик	Intel® Parallel Studio XE 2013
Библиотеки OpenCV	Версия 2.4.3
Математический пакет	MATLAB R2010a
Visual Object Challenge Development Kit	Версия VOCdevkit_08-Jun-2007

СОДЕРЖАТЕЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ ДЕТЕКТИРОВАНИЯ ОБЪЕКТОВ

Постановка задачи

- Рассматривается задача детектирования объектов разных ***классов*** (машина, птица, человек и т. п.) на изображениях или кадрах видео.
- Требуется выполнить поиск объектов некоторого класса на изображении (сцена может содержать несколько объектов одного класса).
- ***Положение*** объекта определяется прямоугольником, окаймляющим его границы.
- Необходимо построить на изображении множество окаймляющих прямоугольников для объектов заданного класса.

МАТЕМАТИЧЕСКАЯ ПОСТАНОВКА ЗАДАЧИ ПОИСКА ОБЪЕКТОВ С *N* ЧАСТЯМИ

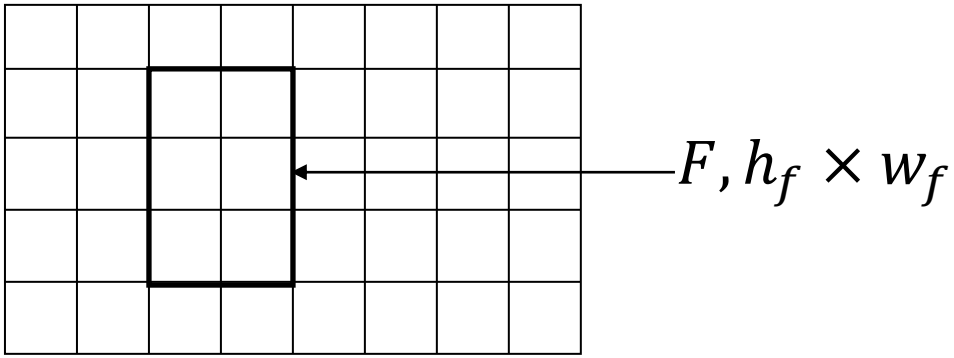
Математическая модель изображения

- Пусть дано изображение I.
- Изображение представляется в виде матрицы пикселей.
- Каждый элемент матрицы состоит из трех компонент. Компоненты соответствуют интенсивностям цвета (RGB) в точке.
- Всего в изображении w элементов по горизонтали и h элементов по вертикали.

- Математической моделью изображения является ***карта признаков (свойств)***. Карта признаков представляет матрицу вещественных чисел, полученных в результате вычисления некоторой функции от интенсивностей текущего и набора окрестных пикселей.

Математическая модель объекта

- Пусть по изображению построена карта свойств $G(i,j)$, и имеется объект для поиска.
- Искомый объект можно описать с помощью фильтра $F = \{f_{x',y'}, x' \in \{0,\dots,w_f\}, y' \in \{0,\dots,h_f\}\}$, где h_f и w_f – размеры фильтра изображения.
- ***Фильтр*** определяет набор признаков, наиболее характерных для объекта заданного класса.



Построение оценочной функции положения объекта (1)

- Для оценки наличия объекта в конкретной области применяется следующая свертка:

$$S(x,y) = \sum_{x',y'} F(x',y')G(x+x',y+y'),$$

где $x' \in \{0,\dots,w_f\}, y' \in \{0,\dots,h_f\}, 0 \le x \le \left\lfloor \frac{w-1}{k} \right\rfloor, 0 \le y \le \left\lfloor \frac{h-1}{k} \right\rfloor$.

- Чем больше $S(x,y)$, тем больше вероятность того, что искомый объект находится в точке (x,y).

Построение оценочной функции положения объекта (2)

- Теперь предположим, что объект состоит из n частей.

- Введем обозначения:
  - F_0 – ***грубый фильтр*** (фильтр для всего объекта),
  - F_i – фильтр для i –ой части объекта (***точный фильтр***). Заметим, что положение точного фильтра задается относительно грубого.

Построение оценочной функции положения объекта (3)

- В этом случае оценочную функцию можно записать следующим образом:

$$S(x,y) = \sum_{i=0}^{n} \sum_{x_i', y_i'} F_i(x_i', y_i')\, G(x_i(x) + x_i', y_i(x) + y_i'),$$

где $x_i(x)$ и $y_i(y)$ – положение фильтра F_i в глобальных координатах изображения, а $x_i' \in \{0, \dots, w_{fi}\}, y_i' \in \{0, \dots, h_{fi}\}$, h_{fi} и w_{fi} – размеры точного фильтра с номером i.

Построение оценочной функции положения объекта (4)

- $S(x, y)$ позволяет найти на изображении объект фиксированного размера.
- Если объект имеет размеры, отличные от эталонного, то в этом случае, объект не будет найден.
- Решение – построение ***пирамиды признаков*** H:
  - Пирамида признаков содержит несколько уровней, на каждом из которых находится карта свойств изображения, полученного в результате уменьшения или увеличения исходной картинки.
  - На уровне λ находится карта свойств исходного изображения, а на 0-ом уровне – карта свойств изображения, которое увеличено в два раза.

Построение оценочной функции положения объекта (5)

- Далее предполагается, что фильтр может быть расположен на любом уровне пирамиды признаков.
- Пусть $p_i = (x_i, y_i, l_i)$ – положение фильтра F_i на уровне в пирамиде признаков H.
- $\varphi(H, p_i)$ получается из пирамиды признаков H и координат положения p_i конкретное свойство изображения путем вычисления глобальных координат x и y фильтра F_i на слое l.
- Тогда оценочную функцию можно записать следующим образом:

$$S(x, y) = \sum_{i=0}^{n} F_i \varphi(H, p_i)$$

Построение оценочной функции положения объекта (6)

- Все части изображения были никак не связаны.
- Для реального объекта это не так!
- Пусть заданы модели частей объекта $P_i = (F_i, V_i, d_i)$, где V_i – идеальное расположение его части, а $d_i \in R^4$ – коэффициенты квадратичной функции штрафа $d_1 x + d_2 y + d_3 x^2 + d_4 y^2$, которая вносит вклад в значение оценочной функции в случае чрезмерного удаления части от самого объекта.
- Модель для объекта с n частями формально определяется множеством параметров:

$$(F_0, P_1, \dots, P_n, b)$$

где параметр b определяет соответствие коэффициентов между моделями

Построение оценочной функции положения объекта (7)

$$score(p_0, p_1, \dots, p_n) = \sum_{i=0}^{n} F_i' \varphi(H, p_i) - \sum_{i=1}^{n} d_i \varphi_d(dx_i, dy_i) + b,$$

первое слагаемое – результат применения фильтров к исходному изображению (значения сверток грубого и точных фильтров с конкретной матрицей признаков),

второе слагаемое – штраф за счет деформации взаимного расположения частей,

$\boldsymbol{b}$ – параметр соответствия коэффициентов между моделями.

Построение оценочной функции положения объекта (8)

$$score(p_0, p_1, \dots, p_n) = \big(\beta, \psi(H, z)\big),$$

где $\beta = \big(F_0', F_1', \dots F_n', d_{1,} \dots d_n, b\big)$,

$\psi(H, z)$

$= (\varphi(H, p_0), \dots, \varphi(H, p_n), -\varphi_d(dx_1, dy_1), \dots, -\varphi_d(dx_n, dy_n), 1)$

Поиск частично видимых объектов

- Для определения положения частично видимых объектов при вычислении сверток каждую матрицу признаков в пирамиде H необходимо дополнить нулевыми границами.
- Размер границы определяется максимальными размерами фильтров по каждому измерению согласно формулам:

$$bx = \frac{mwf}{2} + 1, by = \frac{mhf}{2} + 1,$$

где $mwf = max_f\{w_f\}$, $mhf = max_f\{h_f\}$.

- ***Замечание:*** в свертках с грубым фильтром матрица признаков дополняется нулевой границей указанного размера, с точными фильтрами данная граница должна быть удвоена.

Приведение задачи вычисления значений оценочной функции к задаче оптимизации (1)

$$score(p_0) = \max_{p_1,\dots,p_n} \{score(p_0, p_1, \dots, p_n)\}$$

- Некоторые элементарные преобразования:

$$F_0'\varphi(H, p_0) + \max_{p_1,\dots,p_n} \left\{ \sum_{i=1}^{n} \left(F_i'\varphi(H, p_i) - d_i\varphi_d(dx_i, dy_i) \right) \right\} + b$$

$$= F_0'\varphi(H, p_0) + \sum_{i=1}^{n} \max_{p_1,\dots,p_n} \{F_i'\varphi(H, p_i) - d_i\varphi_d(dx_i, dy_i)\} + b$$

- Далее введем вспомогательные обозначения:

$$R_{0,l_0}(x_0, y_0) = F_0{}'\varphi(H, p_0)$$

$$R_{i,l}(x, y) = F_i{}'\varphi\big(H, (x, y, l)\big)$$

$$D_{i,l}(x, y) = \max_{dx,dy} \big\{ R_{i,l}(x + dx, y + dy) - d_i\varphi_d(dx, dy) \big\}$$

Приведение задачи вычисления значений оценочной функции к задаче оптимизации (2)

$$score(p_0) = score(x_0, y_0, l_0)$$
$$= R_{0,l_0} + \sum_{i=1}^{n} D_{i,l_0-\lambda}(2(x_0, y_0) + V_i) + b$$

- Для вычисления оценки необходимо решить несколько задач оптимизации вида:

$$\max_{dx,dy}\{R_{i,l}(x + dx, y + dy) - d_i\varphi_d(dx, dy)\}$$
$$\max_{dx,dy}\{R_{i,l}(x + dx, y + dy) - d_i\varphi_d(dx, dy)\} =$$
$$= \min_{dx,dy}\{-R_{i,l}(x + dx, y + dy) + d_i\varphi_d(dx, dy)\} =$$
$$= \min_{x',y'}\{-R_{i,l}(x', y') + d_i\varphi_d(x' - x, y' - y)\}$$

где $x' = x + dx, y' = y + dy.$

Приведение задачи вычисления значений оценочной функции к задаче оптимизации (3)

- Очевидно, что при $d_i = (0,0,1,1)$ функция $d_i\varphi_d(x' - x, y' - y)$ представляет собой Евклидово расстояние, т.к. $\varphi_d(dx, dy) = (dx, dy, dx^2, dy^2)$:

$$d_i\varphi_d(x' - x, y' - y) = (x' - x)^2 + (y' - y)^2$$
$$= (x - x')^2 + (y - y')^2$$

- Как следствие, получаем задачу минимизации выпуклой функции, которая решается с использованием ***обобщенного метода преобразования расстояний***.

Приведение задачи вычисления значений оценочной функции к задаче оптимизации (4)

- Задача выбора наиболее вероятных положений объекта предполагает выбор тех положений грубого фильтра, для которых выполняется условие.

$$score(p_0) > threshold,$$

где $threshold$ – некоторое пороговое значение оценочной функции, которое является параметром модели.

- Поиск положений, удовлетворяющих указанному условию, осуществляется в результате прохода по всем уровня пирамиды.
- В результате останется только пересчитать координаты прямоугольников, полученных на каждом уровне, в координаты исходного изображения.

ОСНОВНЫЕ ЭТАПЫ РЕШЕНИЯ ЗАДАЧИ ДЕТЕКТИРОВАНИЯ С ПОМОЩЬЮ LATENT SVM

Построение пирамиды признаков

- ***Построение пирамиды признаков*** – модели исходного изображения. Процедура включает следующие действия:
  - Масштабирование исходного изображения. Результатом является *пирамида изображений*.
  - Построение матриц векторов признаков для каждого изображения в пирамиде изображений – формирование пирамиды признаков.
- Процедура построения пирамиды признаков в настоящей работе не описывается, т.к. время выполнения данного этапа вносит незначительный вклад в общее время поиска объектов на наборе тестовых данных.
- ***Математическая модель объекта*** – грубый фильтр и набор точных фильтров (описана ранее).

Определение положения объекта

- Вычисление значений оценочной функции для каждого возможного положения объекта, исходя из формул для вычисления $score(x_0, y_0, l_0)$. Оценочная функция – сумма скалярных произведений векторов (***свертка***) признаков грубого и точных фильтров модели с соответствующими векторами матрицы признаков. На каждом уровне пирамиды вычисления проводятся независимо.

- Выбор положений, для которых значения оценочной функции превышают пороговое значение.

- Преобразование координат, соответствующих найденным положениям объекта на различных уровнях пирамиды.

ОБЗОР БАЗОВОЙ ПРОГРАММНОЙ РЕАЛИЗАЦИИ АЛГОРИТМА LATENT SVM

Основные структуры данных (1)

- **position** – структура, предназначенная для хранения положения объекта:
  - **unsigned int x** – координата по горизонтали,
  - **unsigned int y** – координата по вертикали,
  - **unsigned int l** – уровень пирамиды признаков.

Основные структуры данных (2)

- **filterObject** – структура для хранения параметров фильтра (используется для хранения грубых и точных фильтров):
  - **position V** – расположение точного фильтра относительно грубого,
  - **float fineFunction[4]** – коэффициенты штрафной функции (порядок коэффициентов соответствует функции $d_1 x + d_2 y + d_3 x^2 + d_4 y^2$),
  - **unsigned int sizeX** – размерность по горизонтали,
  - **unsigned int sizeY** – размерность по вертикали,
  - **unsigned int p** – размерность вектора признаков (=31),
  - **float *H** – матрица весов фильтра (по строкам).

Основные структуры данных (3)

- **featureMap** – структура хранения матрицы признаков на одном уровне пирамиды:
  - **int sizeX** – размерность матрицы признаков по горизонтали,
  - **int sizeY** – размерность матрицы признаков по вертикали,
  - **int p** – размерность вектора признаков (всегда принимает значение, равное 31),
  - **float *map** – матрица признаков, развернутая по строкам.

Основные структуры данных (4)

- **featurePyramid** – структура хранения пирамиды признаков:
  - **int countLevel** – количество уровней в пирамиде признаков,
  - **int lambda** – коэффициент масштабирования (параметр λ в математической модели),
  - **featureMap **pyramid** – массив матриц признаков для всех уровней пирамиды изображений.

Структура программных модулей

- В реализации алгоритма вывода выделено два программных модуля:
  - Модуль построения пирамиды признаков (**featurePyramid**).
  - Модуль вычисления положения объекта заданного класса (**matching**).

Функции модуля построения пирамиды признаков (1)

- Модуль построения пирамиды признаков поддерживает следующую функциональность:
  - Масштабирование исходного изображения. Данная процедура позволяет получить набор изображений с увеличенным и уменьшенным разрешением – пирамиду изображений.
  - Построение матриц векторов признаков (карты свойств) для каждого изображения в пирамиде – **getFeaturePyramid**. Указанная функция включает следующие этапы:
    - Разбиение изображения на блоки фиксированного размера. Блок состоит из подмножества пикселей, каждый из которых характеризуется интенсивностью цвета.

Функции модуля построения пирамиды признаков (2)

- Построение гистограммы градиентов по изображению.
- Построение объединенной гистограммы в ячейки (исходной карты свойств) – **getFeatureMaps_dp**.
- Нормализация и отсечение карты свойств – **normalizationAndTruncationFeatureMaps**.
- Снижение размерности карты свойств – **PCAFeatureMaps**.

Функции модуля поиска положений (1)

- Оценка всех возможных положений объекта на изображении (построение значений оценочной функции) – **thresholdFunctionalScore**.
- Для получения значений оценочной функции используется программная функция **convolution**, которая отвечает за вычисление значений свертки матрицы признаков и фильтра по определению.

Функции модуля поиска положений (2)

- Для вычисления сверток с использованием БПФ разработан набор функций, необходимых с точки зрения алгоритма:
  - **fftImagesMulti** – функция вычисления произведения образов Фурье,
  - **rot2PI** – функция поворота матрицы фильтра,
  - **getFFTImageFilterObject** – функция вычисления образа Фурье для фильтра,
  - **getFFTImageFeatureMap** – функция вычисления образа Фурье для матрицы признаков,
  - **convFFTConv2d** – функция вычисления свертки средствами БПФ.

Функции модуля поиска положений (3)

- Выделение наиболее вероятных положений объекта.
- Построение прямоугольников, окаймляющих объект, на основании полученных положений.
- Отсечение частей прямоугольников, выходящих за границы изображения, – **nonMaximumSuppression**.
- Отображение изображения и окаймляющих прямоугольников средствами OpenCV – **showBoxes**.

Вычисление сверток

ОПТИМИЗАЦИЯ И РАСПАРАЛЛЕЛИВАНИЕ БПФ

Анализ последовательной реализации

- Провести анализ эффективности последовательной реализация, в которой для вычисления сверток матриц векторов признаков с фильтрами используется двумерное БПФ. Для этого необходимо воспользоваться инструментом Intel Parallel Amplifier в режиме “Hotspots”.
- Результаты такого анализа позволят установить, что функция вычисления одномерного БПФ (**fft**) выполняется примерно 95% времени от общего времени поиска, поэтому ее необходимо оптимизировать.
- Просмотр более детальной трассы покажет, что наибольшее время тратится на вычисление синусов и косинусов. Двумерное быстрое преобразование Фурье реализуется через одномерное.

Оптимизация БПФ

- Так как изменение аргументов функций синуса и косинуса подчиняется определенному закону, подсчет каждого последующего значения тригонометрической функции можно выполнить на основании предыдущего значения.
- Величина угла на каждой итерации изменяется по закону:

$$\gamma_i = \gamma_0 \cdot i$$

- Таким образом, вычисление синуса и косинуса сводится к однократному их вычислению, $\sin\gamma_0, \cos\gamma_0$, а последующие значения вычисляются за счет уже имеющихся данных с помощью простых тригонометрических формул:

$$\sin\gamma_{i+1} = \sin(\gamma_i + \gamma_0) = \sin\gamma_i \cos\gamma_0 + \cos\gamma_i \sin\gamma_0$$
$$\cos\gamma_{i+1} = \cos(\gamma_i + \gamma_0) = \cos\gamma_i \cos\gamma_0 - \sin\gamma_i \sin\gamma_0$$

Распараллеливание одномерного БПФ

- Выполнить распараллеливание рекурсивной реализации одномерного БПФ с помощью механизма задач (task) библиотеки TBB.
- Оценить масштабируемость разработанной параллельной реализации. Для этого необходимо провести эксперименты по запуску параллельной реализации в разное количество потоков на некотором наборе изображений.

Распараллеливание двумерного БПФ

- Вернуться к последовательной оптимизированной версии. Выполнить распараллеливание двумерного БПФ.
- Оценить масштабируемость разработанной параллельной реализации.
- Провести анализ проделанной работы.

Вычисление сверток

ОПТИМИЗАЦИЯ И РАСПАРАЛЛЕЛИВАНИЕ ВЛОЖЕННЫХ ЦИКЛОВ

Анализ эффективности последовательной реализации

- Провести анализ эффективности последовательной реализации, в которой для вычисления сверток матриц векторов используется набор вложенных циклов.
- Как и на предыдущем этапе предлагается воспользоваться инструментом Intel Parallel Amplifier в режиме “Hotspots”.
- Результаты такого анализа должны выявить, что наиболее трудоемкой функцией является функция вычисления сверток – **convolution**.

Упрощение и разворачивание циклов

- Выполнить упрощение вложенного цикла – добавление результата очередного умножения осуществлять во временную переменную, результирующее значение которой сохранить в массив сверток. Оценить прирост производительности.
- Выполнить разворачивание внутреннего цикла. Провести эксперименты с разворачиванием на 2, 3, 4 и 5 итераций. Оценить эффективность работы приложения для каждого возможного значения развертки.

Распараллеливание сверток

- Выполнить распараллеливание оптимизированной версии сверток с использованием OpenMP. Подобрать оптимальное значение размера порции при формировании расписания.
- Провести анализ масштабируемости разработанной параллельной реализации.
- Выполнить распараллеливание последовательной оптимизированной версии сверток с использованием средств библиотеки TBB. Подобрать оптимальное значение размера порции при формировании расписания.
- Провести исследование масштабируемости разработанной параллельной реализации.

Сравнение реализаций

- Выполнить анализ результатов, полученных при распараллеливании сверток, которые вычислялись с использованием БПФ и напрямую через набор вложенных циклов.

РАСПАРАЛЛЕЛИВАНИЕ ВЫЧИСЛЕНИЙ ПО УРОВНЯМ ПИРАМИДЫ ПРИЗНАКОВ

Анализ степени параллелизма предыдущей параллельной реализации

- Оценим степень параллелизма реализации, использующей параллельное вычисление сверток через набор вложенных циклов.
- Данная реализация была разработана на предыдущем шаге оптимизации и распараллеливания.
- Для оценки степени параллелизма обратимся к результатам Concurrency-анализа, которые можно получить с помощью Intel VTune Amplifier XE.
- Обратите внимание на диаграмму распределения нагрузки между потоками, из нее видно, что потоки периодически возобновляются, когда необходимо вычислить свертки, и засыпают, когда в основном потоке выполняется другая работа.

Распараллеливание вычислений по уровням пирамиды признаков (1)

- Вернуться к последовательной оптимизированной реализации, в которой свертки вычисляются с использованием вложенных циклов.
- Выполнить распараллеливание цикла построения оценочной функции на уровнях пирамиды признаков. На данном этапе предлагается использовать динамическое распределение нагрузки между потоками.

Распараллеливание вычислений по уровням пирамиды признаков (2)

- Вернуться к последовательной оптимизированной реализации, в которой свертки вычисляются с использованием вложенных циклов.
- Выполнить распараллеливание цикла построения оценочной функции на уровнях пирамиды признаков с использованием статического расписания.
- Для этого необходимо разработать функцию распределения уровней пирамиды признаков между потоками с точки зрения числа операций. Можно считать, что общее число операций, выполняемых при оценке возможных положений на уровне пирамиды признаков, определяется исключительно числом умножений и сложений, которые выполняются при вычислении сверток.

Распараллеливание вычислений по уровням пирамиды признаков (3)

- Сравнить времена работы приложений при использовании динамического и статического расписания. Выбрать наиболее эффективное.
- Провести анализ масштабируемости более эффективной параллельной реализации. Объяснить причины отсутствия линейного ускорения.

ОЦЕНКА КАЧЕСТВА ПОИСКА ОБЪЕКТОВ РАЗНЫХ КЛАССОВ

Тестовая база изображений

- Предлагается выполнить массовые эксперименты на базе PASCAL VOC 2007 и оценить качество детектирования Latent SVM на каждом классе объектов, входящих в состав VOC.
- База данных содержит изображения объектов двадцати классов (aeroplane, bicycle, bird, bottle и др.).
- Представленные фотографии различаются размером изображенных на них объектов, их положением на сцене, ракурсом и степенью освещенности. Указанные факторы оказывают значительное влияние на точность построенной модели.

Средняя точность предсказания

- Средняя точность предсказания (average precision):

$$AP = \frac{1}{11} \sum_{r \in \{0;0.1;\ldots;1\}} p(r), \qquad p(r) = \max_{\bar{r}, \bar{r} \geq r} p(\bar{r})$$

где $p(\bar{r}) = \frac{a}{a+c}$ – точность, $\bar{r}$– процент перекрытия детектированного окаймляющего прямоугольника и прямоугольника, который был размечен на исходном изображении как окаймляющий $\bar{r} \in \{0; 0.1; \ldots; 1\}$,
a – количество объектов, для которых процент перекрытия не меньше, чем $\bar{r}$ (т. е. считается, что объект детектирован правильно),
c– количество объектов с процентом перекрытия, меньшим, чем $\bar{r}$ (объект найден ошибочно).

Инструмент для вычисления средней точности предсказания

- Для вычисления средней точности на основании информации о продетектированных прямоугольниках предлагается воспользоваться инструментом VOC Development Kit.
- Инструкция по работе с VOCdevkit находится в приложении (Приложение В. Инструкция по запуску пакета VOCdevkit).

Дополнительные задания (1)

- Разработайте и встройте реализацию функции вычисления сверток с использованием NVIDIA CUDA SDK. Оцените эффективность разработанной реализации по сравнению с реализациями, предложенными в работе.
- Разработайте и встройте реализацию функции вычисления сверток с использованием Intel® OpenCL SDK. Оцените эффективность разработанной реализации по сравнению с реализациями, предложенными в работе.

Дополнительные задания (2)

- Разработайте и встройте реализацию функции вычисления сверток с использованием быстрого преобразования Фурье, имеющегося в библиотеке Intel® Math Kernel Library. Оцените эффективность разработанной реализации по сравнению с реализациями, предложенными в работе.
- Разработайте параллельную реализацию функции вычисления сверток, использующей вложенные циклы, с помощью Intel® Cilk Plus. Оцените эффективность разработанной реализации по сравнению с реализациями, предложенными в работе.

Авторский коллектив

- Козинов Евгений Александрович,
  ассистент кафедры
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  evgeniy.kozinov@gmail.com
- Кустикова Валентина Дмитриевна,
  ассистент кафедры
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  valentina.kustikova@gmail.com
- Половинкин Алексей Николаевич,
  младший научный сотрудник
  alexey.polovinkin@gmail.com