Подходы к оптимизации и распараллеливанию вычислений в задаче детектирования объектов разных классов на изображении*

Е.А. Козинов, В.Д. Кустикова, И.Б. Мееров, А.Н. Половинкин, А.А. Сиднев

Нижегородский государственный университет им. Н.И. Лобачевского

В работе рассматривается задача детектирования объектов разных классов на статических изображениях. Описывается схема решения данной задачи с использованием алгоритма Latent SVM. Используется известный подход к ускорению вычислений – построение каскада классификаторов. Обсуждаются проблемы распараллеливания и оптимизации времени расчета. Проводится анализ вариантов решения указанных проблем. Даются результаты вычислительных экспериментов, формулируются выводы и планы по дальнейшему развитию.

1. Введение

В настоящее время при решении многих практических задач используются системы компьютерного зрения (системы видеонаблюдения, управления процессами, организации информации и др.). Ядром вычислений в таких системах является обработка видеоданных. Поскольку основное применение результатов компьютерного зрения сосредоточено в области промышленной робототехники (автономные роботы, системы визуального контроля и измерений), то ключевым становится не только качество, но и скорость обработки видеоданных. Центральная задача, которая решается многими исследовательскими группами, – это анализ качественного состава сцены. При этом исследования ведутся как в направлении повышения точности распознавания объектов на кадрах видеопотока, так и в направлении снижения времени обработки видеоинформации.

В работе рассматривается задача детектирования объектов разных классов (автомобили, автобусы, люди и др.) на изображениях с использованием алгоритма поиска Latent SVM [4]. Приводится вычислительная схема решения данной задачи. Описывается один из наиболее общих алгоритмических подходов к ускорению вычислений в задачах компьютерного зрения – использование каскада классификаторов – на примере модификации Latent SVM [5]. Обсуждаются проблемы оптимизации и распараллеливания вычислений в системах с общей памятью. Выполняется анализ различных вариантов решения указанных проблем, предлагаются результаты вычислительных экспериментов на базе изображений PASCAL Visual Object Challenge 2007 (VOC2007, http://pascallin.ecs.soton.ac.uk/challenges/VOC). Разработка выполнена в рамках широко известной библиотеки компьютерного зрения OpenCV (http://sourceforge.net/projects/opencvlibrary).

2. Задача детектирования объектов на изображении с использованием методов, основанных на извлечении признаков

2.1 Постановка задачи

Задача детектирования объектов состоит в том, чтобы определить наличие объекта на изображении и найти его положение в системе координат пикселей исходного изображения. Положение объекта в зависимости от выбора алгоритма детектирования может определяться координатами прямоугольника, окаймляющего объект, либо контуром этого объекта, либо

* Работа выполнена в рамках программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы», государственный контракт № 11.519.11.4015.

координатами набора точек, наиболее характерных для объекта. В данной работе считается, что положение определяется координатами окаймляющего прямоугольника.

Все вычисления должны выполняться в реальном времени, поэтому основная цель настоящей работы состоит в том, чтобы максимально уменьшить среднее время поиска за счет оптимизации и распараллеливания вычислений.

2.2 Общая схема решения

Один из возможных подходов к решению задачи детектирования состоит в том, чтобы использовать алгоритмы машинного обучения для построения моделей классов объектов и алгоритмы вывода для поиска объектов на изображении.

Построение модели состоит из двух этапов:

1. Извлечение признаков, характерных для объектов класса, – построение характеристических векторов-признаков для особых точек объекта (углов, линий, ребер [15] или контуров объектов [16]) или для всего объекта.
2. Построение модели объекта на полученных признаках для последующего распознавания объектов. На данном этапе выполняется тренировка какого-либо классификатора.

Методы, основанные на извлечении признаков, описывают объект с использованием векторов-признаков. Вектора строятся на основании цветовой информации (гистограмма ориентированных градиентов (Histogram of Oriented Gradients или HOG) – один из наиболее популярных способов). Также может быть использована контекстная информация (context based) [17,18], а в некоторых случаях – данные о геометрии и взаимном расположении частей объекта (part-based) [4]. Тем не менее, все эти методы строят некоторую математическую модель объекта на каждом изображении тренировочной выборки, содержащем объект. Формально признак является числовой характеристикой. Таким образом, объект описывается набором векторов признаков в характерных точках. В результате тренировки строится модель, содержащая «усредненные» вектора признаков.

Алгоритм вывода (поиска) по существу включает два этапа:

1. Извлечение признаков объекта из тестового изображения согласно алгоритму, который использовался в процессе тренировки. При извлечении признаков возникает две основные проблемы:
   - на изображении может быть много объектов одного класса, а необходимо найти всех представителей. Поэтому необходимо просматривать все части изображения, проходя «бегущим» окном (sliding window) от левого верхнего до правого нижнего угла. При этом размер окна определяется размером изображений тренировочной выборки;
   - объекты на изображении могут иметь разный масштаб. Самое распространенное решение – масштабирование изображения и построение пирамиды изображений.
2. Поиск объектов на изображении. Входными данными этапа поиска являются формальное описание объекта – набор признаков, которые выделены из тестового изображения, – и модель класса объектов. На основании этой информации классификатор принимает решение о принадлежности объекта классу. Некоторые методы поиска также оценивают степень достоверности того, что объект принадлежит рассматриваемому классу.

Качество методов в основном зависит от того, насколько хорошо выбраны признаки, т.е. насколько хорошо эти признаки дифференцируют классы объектов. Существуют специализированные методы, основанные на извлечении признаков, например, для детектирования лиц [7, 8, 10], транспортных средств [11] и пешеходов [12, 13, 14].

3. Поиск объектов с использованием алгоритма Latent SVM

3.1 Модель объектов некоторого класса

При построении алгоритма Latent SVM модель объекта некоторого класса определяется набором компонент, каждая из которых соответствует одному из ракурсов этого объекта. Компонента включает совокупность фильтров [4]:

- грубый фильтр, определяющий набор векторов признаков, наиболее характерных для всего объекта;
- совокупность точных фильтров, описывающих отдельные части объекта.

3.2 Вычислительная схема поиска объектов

Алгоритм поиска Latent SVM состоит из двух этапов:

1. Построение пирамиды признаков.
2. Определение наиболее вероятных положений объектов заданного класса.

Построение пирамиды признаков включает масштабирование изображений – формирование пирамиды изображений [1, 4] – и построение матриц векторов-признаков для каждого изображения в пирамиде. Вектор признаков представляет гистограмму ориентации градиентов (Histogram of Oriented Gradients (HOG), [4]) для некоторого блока пикселей исходного изображения.

Смысл этапа поиска положения объекта состоит в том, чтобы некоторым образом оценить вероятность нахождения объекта во всех возможных положениях пирамиды признаков и выбрать среди всего множества наиболее вероятное. Для определения положения объекта в алгоритмах [4, 5] вычисляется значение оценочной функции для каждого возможного положения. Расположение объекта определяется положением левого верхнего угла грубого фильтра в наборе векторов матрицы признаков какого-либо уровня. Оценочная функция строится как сумма скалярных произведений наборов векторов признаков грубого и точных фильтров модели с соответствующими наборами векторов признаков, построенных на изображении. В дальнейшем сумму скалярных произведений векторов фильтра и изображения будем называть сверткой. Дополнительно оценочная функция содержит слагаемое, которое определяет штраф за чрезмерное удаление частей объекта от самого объекта. Коэффициенты функции штрафа являются параметрами компоненты модели.

4. Каскад классификаторов

4.1 Схема построения каскада классификаторов

Построение и использование каскада классификаторов – один из наиболее распространенных подходов [2, 3] к ускорению вычислений при решении различных задач классификации. По существу идея построения аналогична AdaBoost-классификатору (adaptive boost, [6]). На каждой стадии тренировки строится некоторый классификатор, при этом учитывается информация об ошибках предшествующего классификатора. Итоговый классификатор представляется комбинацией построенных классификаторов.

При решении задач детектирования и классификации объектов на изображениях последовательность классификаторов, полученных на этапе тренировки, сортируется в зависимости от величины ошибки. Применение первых более слабых классификаторов позволяет последовательно отсекать положения, в которых заведомо не может быть обнаружен объект. Таким образом, сильные классификаторы (как правило, вычислительно более трудоемкие) применяются не ко всем возможным положениям объекта, а только к некоторому небольшому набору.

Каскад классификаторов нашел применение при решении многих задач компьютерного зрения. Наиболее известным алгоритмом, в основе которого лежит каскад классификаторов, является алгоритм Виолы-Джонса для детектирования лиц [7, 8]. Идея построения каскада также используется при реализации каскадного преобразования Хафа [9], которое позволяет искать структурные объекты на изображении (линии, окружности и т.п.), и многих других алгоритмов. Далее рассмотрим каскадную схему алгоритма Latent SVM, предназначенного для детектирования объектов разных классов.

4.2 Детектирование объектов с использованием каскадного Latent SVM

4.2.1 Модель объектов некоторого класса

В каскадном Latent SVM [5] в модель объектов каждого класса включается дополнительная информация:

- PCA-проекции грубого и точных фильтров в пространство меньшей размерности;
- по два порога для каждого фильтра и его проекции:
  - порог идентификации присутствия объекта или его части, который определяет возможность присутствия объекта или его части в определенном месте изображения;
  - порог гипотетического отклонения определяет, насколько часть объекта может быть смещена от своего идеального расположения относительно грубого фильтра.

4.2.2 Вычислительная схема поиска объектов

Каскадный алгоритм Latent SVM [5] включает два этапа каскадной схемы, на которых выполняются следующие действия:

1. Построение множества возможных положений объекта. Для этого применяется преобразованная дополнительная пирамида признаков и соответствующие ей фильтры. На данном этапе осуществляется:

   1.1. Вычисление свертки грубого фильтра (PCA) в первом возможном положении (например, в левом верхнем углу изображения на одном из уровней). Если свертка меньше порога идентификации для грубого фильтра, то необходимо выполнить переход к рассмотрению следующего возможного положения грубого фильтра.

   1.2. Перебор сверток точных фильтров.

   1.2.1. Выбор первого фильтра и вычисление его свертки с векторами матрицы признаков в некоторой окрестности грубого фильтра.

   1.2.2. Проверка условия: если сумма свертки точного фильтра и текущей оценки положения объекта меньше порога гипотетического отклонения, то положение грубого фильтра не рассматривается.

   1.2.3. Выбор положения с максимальной суммой текущей оценки и свертки среди всех подсчитанных положений. Сумма принимается за текущую оценку положения объекта.

   1.2.4. Проверка условия: если полученная оценка положения объекта меньше порога деформации для текущего точного фильтра, то рассматривается следующее положение грубого фильтра, иначе анализируется положение следующего точного фильтра.

   1.3. Если для всех точных фильтров было найдено расположение, то положение грубого фильтра считается возможным положением объекта на изображении текущего масштаба.
2. Определение наиболее вероятных положений объекта среди множества, построенного на предыдущем шаге. На данном этапе используется полная пирамида признаков. Как правило, количество возможных положений не очень большое. Процесс проверки существования объекта на полной пирамиде производится по той же схеме, что и для свернутой пирамиды признаков.

5. Этапы оптимизации и распараллеливания последовательной реализации алгоритма Latent SVM

5.1 Тестовая инфраструктура

На каждом этапе оптимизации и распараллеливания последовательной реализации Latent SVM в качестве тестового набора использовалась база данных конкурса PASCAL VOC 2007, содержащая различные 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.

Так как исходная задача состояла в реализации алгоритма поиска и не включала в себя реализацию алгоритма обучения, для проведения экспериментов были использованы модели авторов статей [4, 5], преобразованные в формат xml в соответствии с разработанной структурой.

Оценка качества детектирования объектов с помощью реализованного алгоритма вывода выполнялась средствами VOC Development Kit, в состав которого входит модуль, позволяющий вычислить среднюю точность предсказания (average precision). Указанная метрика определяется как математическое ожидание точностей следующим образом:

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

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 $\bar{r} \in [0;0.1;0.2;...;1]$, 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).

Вычислительные эксперименты проводились с использованием следующей инфраструктуры:

- Язык разработки: C.
- Среда разработки: Microsoft Visual Studio 9.0.
- Компилятор: Microsoft C/C++ Compiler Version 15.00.30729 (x64).
- Процессор: 2 двухъядерных процессора Intel Xeon 5150 (2.66 GHz).
- Память: 4 Gb.
- Библиотеки: Intel Threading Building Blocks (TBB) 3.0 for Windows, v.3.0.2010.707.
- Технологии: OpenMP.
- Операционная система: Microsoft Windows Server 2008 Standard SP1 x64.

Далее в работе приводится последовательность шагов оптимизации и распараллеливания. На каждом этапе, который дает выигрыш по отношению к реализациям авторов [4, 5], приводятся результаты экспериментов по средней точности детектирования и среднему времени детектирования объектов на одном изображении. Объем тестовой выборки составляет примерно 5000 изображений.

5.2 «Горячие» точки последовательной реализации

Первый шаг при выполнении оптимизации заключается в локализации «горячих» точек программы. Для этого мы воспользовались инструментом Intel Parallel Amplifier в режиме “Hotspots”. Результат анализа показал, что в разработанной вычислительной схеме метода Latent SVM основная операция – это вычисление сверток. Входными данными операции являются:

1. Двумерная матрица векторов признаков *featureMap* (матрица признаков на некотором уровне пирамиды признаков). Данная матрица может быть представлена в виде трехмерного прямоугольного параллелепипеда размерности $N_y^f N_x^f p$, где N_x^f – число столбцов матрицы векторов, N_y^f - количество строк матрицы векторов, $p = 31$ – размерность вектора признаков. Двумерную матрицу, полученную при каждом фиксированном p, будем называть каналом.
2. Двумерная матрица весовых векторов фильтра *filter* (это может быть точный или грубый фильтр). Данная матрица может быть представлена в виде трехмерного прямоугольного параллелепипеда размерности $N_y^g N_x^g p$, где N_x^g – число столбцов матрицы весовых

векторов, N_y^g - количество строк матрицы весовых векторов, $p = 31$ – размерность вектора весов.

Результатом выполнения операции свертки является двумерная матрица свертки *conv* размерности $\left(N_y^f - N_y^g + 1\right)\left(N_x^f - N_x^g + 1\right)$.

Задача вычисления свертки матрицы векторов признаков и фильтра сводится к тому, чтобы определить значения элементов матрицы свертки в соответствии с формулой:

$$conv[i,j] = \sum_{di=0}^{\left(N_y^g-1\right)} \sum_{dj=0}^{\left(N_x^g-1\right)} \sum_{k=0}^{p-1} filter[di,dj,k] \cdot featureMap[i+di, j+dj, k]$$

По существу с точки зрения программной реализации получается цикл вложенности 5. Чтобы ускорить вычисление сверток, были выполнены некоторые компиляторные оптимизации (упрощение циклов, вынос инвариантов), но значительный выигрыш не был получен, т.к. очевидно, что компилятор выполнял эти оптимизации автоматически. Распараллеливание циклов в функции вычисления сверток не дает выигрыша, т.к. свертки вычисляются многократно, как следствие, возникают значительные накладные расходы на организацию параллелизма (возобновление и остановка потоков). Именно такие результаты были получены в процессе анализа степени параллелизма разработанной реализации. Таким образом, распараллеливание необходимо выполнять на более высоком уровне.

5.3 Распараллеливание по уровням пирамиды признаков

Как выяснилось ранее, основное время занимает процедура поиска положения объектов (второй этап вычислительной схемы) и наиболее трудоемкой операцией является вычисление сверток. Заметим, что поиск объектов «бегущим» окном выполняется на каждом уровне пирамиды признаков независимо. Таким образом, можно выполнить распараллеливание разработанной последовательной реализации по уровням пирамиды. Более того, на каждом уровне можно подсчитать количество выполняемых операций, как следствие, достаточно просто построить статическое расписание распределения нагрузки между потоками. Предлагается использовать следующую схему распределения – каждому потоку отдавать на обработку набор уровней так, чтобы суммарное количество операций было приближенно одинаковым. Такой подход обеспечит равномерность распределения нагрузки между потоками. Описанная схема распараллеливания была реализована с помощью средств библиотеки TBB.

Таблица 1. Средняя точность детектирования объектов на данных VOC 2007

Название класса объектов	Средняя точность			Время работы (с)		Ускорение (отношение времени последовательной к параллельной)
	Последовательная реализация	Реализация авторов [4]	Параллельная реализация в 4 потока	Последовательная реализация	Текущая реализация в 4 потока	
aeroplane	0,278	0,28	0,279	4,205	1,263	3,329
bicycle	0,525	0,544	0,525	4,412	1,303	3,386
bird	0,006	0,007	0,007	3,746	1,174	3,191
boat	0,123	0,145	0,124	3,623	1,157	3,131
bottle	0,26	0,262	0,261	3,438	1,125	3,056
bus	0,409	0,398	0,409	4,435	1,304	3,401
car	0,457	0,463	0,457	3,215	1,079	2,98
cat	0,163	0,16	0,164	5,032	1,402	3,589
chair	0,154	0,163	0,155	4,89	1,374	3,559
cow	0,167	0,167	0,167	4,696	1,355	3,466

diningtable	0,238	0,243	0,239	5,185	1,432	3,621
dog	0,07	0,05	0,07	4,85	1,372	3,535
horse	0,437	0,438	0,437	5,117	1,422	3,598
motorbike	0,384	0,382	0,385	5,12	1,42	3,606
person	0,338	0,342	0,338	3,747	1,188	3,154
pottedplant	0,101	0,079	0,101	3,417	1,117	3,059
sheep	0,166	0,172	0,166	3,13	1,061	2,95
sofa	0,214	0,221	0,215	4,998	1,399	3,573
train	0,331	0,34	0,331	4,208	1,362	3,089
tvmonitor	0,38	0,393	0,381	4,208	1,334	3,154
			Средние значения:	4,283	1,282	3,321

Таблица 1 содержит результаты вычислительных экспериментов, проведенных над разработанной последовательной и параллельной реализациями. Очевидно, что средняя точность детектирования объектов с помощью подготовленных реализаций (второй и четвертый столбцы) приближенно совпадает с точностью, показанной авторами статьи [4] (третий столбец). Отметим, что, несмотря на практически идентичное качество детектирования, последовательная реализация в среднем обрабатывает одно изображение в 2-2.5 раза медленнее, чем реализация авторов алгоритма, которая решает задачу детектирования объектов одного класса на изображении примерно за 2 секунды [4]. Но дальнейшее распараллеливание последовательной версии по уровням пирамиды признаков в среднем дает ускорение в 3,3 раза на четырех потоках (последний столбец таблицы), в результате чего время обработки было уменьшено в среднем до 1,28 секунды. Указанное ускорение – неплохой результат при условии, что распараллелены не все этапы алгоритма, а только поиск объектов, который занимает ориентировочно 80% времени от общего времени поиска объектов заданного класса на изображении.

5.4 Реализация каскадного алгоритма Latent SVM

Следующим шагом оптимизации является реализация каскадной схемы, описанной в предыдущем разделе.

В таблице 2 приведены численные значения средней точности, полученные на каскадной реализации алгоритма вывода Latent SVM. Сравнение с результатами, приведенными в [5], показывает, что практически на всех классах объектов наблюдаются незначительные отклонения. Также, в таблице представлены результаты точности детектирования для выполненной последовательной реализации алгоритма [4] на моделях из [5], содержащих большее количество компонент. Из результатов сравнения можно видеть, что точность детектирования во всех трех реализациях практически совпадает. В таблице 1 полужирным начертанием выделены наилучшие точности детектирования для каждого класса объектов.

Наибольший интерес в данной работе представляет эффект от применения каскадной схемы. Поэтому обратим внимание на ускорение, которое получено по отношению к предшествующей реализации. Максимальное ускорение составляет 8,45 раза, среднее – примерно 6,5 раза. Очевидно, что применение каскадной схеме позволяет значительно уменьшить среднее время детектирования без потери в средней точности.

Таблица 2. Средняя точность детектирования объектов на данных VOC 2007

Название класса объектов	Средняя точность		Время работы (с)		Ускорение (отношение времени каскадной к предыдущей)
	Каскадная реализация	Реализация авторов [5]	Каскадная реализация	Предыдущая реализация	
aeroplane	0,28	0,23	1,37	10,37	7,57
bicycle	**0,57**	0,49	1,38	11,42	8,28
bird	0,09	**0,11**	1,72	10,76	6,26

boat	**0,15**	0,13	2,14	11,2	5,23
bottle	0,24	**0,27**	2	10,19	5,1
bus	0,47	**0,47**	1,46	11,05	7,57
car	**0,54**	0,50	1,78	9,45	5,31
cat	0,16	**0,19**	1,35	10,8	8
chair	**0,20**	0,16	2,28	10,94	4,8
cow	**0,24**	0,23	1,57	10,88	6,93
diningtable	**0,22**	0,11	1,61	11,14	6,92
dog	0,11	**0,12**	1,35	11,41	8,45
horse	**0,56**	0,36	1,54	10,88	7,06
motorbike	**0,45**	0,37	1,29	10,87	8,43
person	**0,41**	0,38	2,82	10,12	3,59
pottedplant	0,12	**0,14**	2,38	9,71	4,08
sheep	0,18	**0,23**	1,73	9,18	5,31
sofa	0,29	0,23	1,52	10,57	6,95
train	**0,44**	0,34	1,42	11,53	8,12
tvmonitor	0,40	**0,40**	1,66	11,35	6,84
Средние значения:			1,72	10,69	6,54

5.5 Оптимизация и распараллеливание каскадного алгоритма Latent SVM

На данный момент необходимо оценить среднее время обработки одного изображения с использованием разработанной каскадной реализации. Первый шаг состоит в том, чтобы определить «горячие» точки в процедуре поиска. Аналогично предыдущей реализации наиболее затратной операцией является операция вычисления сверток. Отметим, что при этом свертки вычисляются не во всех возможных положениях объекта (не на всей матрице признаков), а только на некотором наборе наиболее вероятных, которые остались после отсечения. Соотношение времени построения пирамиды признаков и времени вычисления сверток близко к единице. На данном этапе операция вычисления сверток была переписана с использованием SSE. В результате в среднем выигрыш от такой оптимизации составил 0,1 секунды (четвертый и пятый столбцы таблицы). Далее было выполнено распараллеливание оптимизированной версии.

Таблица 3. Средняя точность детектирования объектов на данных VOC 2007

Название класса объектов	Средняя точность		Время работы (с)				Ускорение	
	Параллельная каскадная реализация	Реализация авторов [5]	Исходная реализация	Последовательная каскадная реализация	Каскадная реализация после оптимизации	Параллельная реализация в 4 потока	Отношение параллельной каскадной к последовательной	Отношение предыдущей к параллельной
aeroplane	0,28	0,23	10,37	1,37	1,37	0,51	2,69	20,33
bicycle	**0,57**	0,49	11,42	1,38	1,30	0,50	2,60	22,84
bird	0,09	**0,11**	10,76	1,72	1,60	0,56	2,86	19,21
boat	**0,15**	0,13	11,2	2,14	1,97	0,77	2,56	14,55
bottle	0,24	**0,27**	10,19	2,00	1,81	0,66	2,74	15,44
bus	0,47	**0,47**	11,05	1,46	1,35	0,55	2,45	20,09
car	**0,54**	0,50	9,45	1,78	1,62	0,60	2,70	15,75
cat	0,16	**0,19**	10,8	1,35	1,29	0,53	2,43	20,38
chair	**0,20**	0,16	10,94	2,28	2,08	0,75	2,77	14,59
cow	**0,24**	0,23	10,88	1,57	1,47	0,55	2,67	19,78

diningtable	**0,22**	0,11	11,14	1,61	1,49	0,58	2,57	19,21
dog	0,11	**0,12**	11,41	1,35	1,31	0,50	2,62	22,82
horse	**0,56**	0,36	10,88	1,54	1,44	0,54	2,67	20,15
motorbike	**0,45**	0,37	10,87	1,29	1,23	0,49	2,51	22,18
person	**0,41**	0,38	10,12	2,82	2,55	0,86	2,97	11,77
pottedplant	0,12	**0,14**	9,71	2,38	2,17	0,72	3,01	13,49
sheep	0,18	**0,23**	9,18	1,73	1,57	0,59	2,66	15,56
sofa	0,29	0,23	10,57	1,52	1,43	0,55	2,60	19,22
train	**0,44**	0,34	11,53	1,42	1,33	0,53	2,51	21,75
tvmonitor	0,40	**0,40**	11,35	1,66	1,62	0,60	2,70	18,92
Средние значения:			10,69	1,72	10,69	0,597	2,66	18,40

Обратимся к анализу каскадной схемы и возможности применения распараллеливания по уровням пирамиды признаков. На первом этапе каскадной схемы отсекаются наименее вероятные положения объектов. Как следствие, при переходе на следующий этап на каждом уровне пирамиды останутся достоверные положения объекта, количество которых заранее не известно. При этом невозможно спрогнозировать, в какой момент произойдет отсечение (отбрасывание) следующего положения, а потому нельзя вычислить количество выполняемых операций. Реализация такой схемы распараллеливания не даст выигрыша, т.к. невозможно достигнуть равномерного распределения нагрузки между потоками. Поэтому было принято решение параллелить каскадную схему на уровне компонент модели объектов класса. В таблице 3 показаны результаты экспериментов с параллельной реализацией каскадного Latent SVM, выполненной с использованием технологии OpenMP. Разработанная реализация не уступает по точности детектирования реализации авторов [5], а на некоторых классах объектов, напротив, показывает более высокие результаты (выделены полужирным начертанием во втором столбце). В среднем ускорение составило 2,66 раза на четырех потоках. Отметим, что указанная величина во многом определяется количеством возможных положений, которые попадают на вход второго этапа каскадной схемы в каждой компоненте. Таким образом, ускорение может принимать недетерминированные значения. В результате распараллеливания оптимизированной версии каскада в среднем время поиска уменьшилось в 18,4 раза по сравнению с исходной последовательной реализацией (последний столбец таблицы 3).

6. Заключение

В ходе работы выполнены последовательная и параллельная реализации алгоритма вывода Latent SVM [4], позволяющие детектировать объекты с точностью, которая не уступает реализации разработчиков алгоритма, о чем свидетельствуют результаты проведенных экспериментов. Реализации интегрированы в библиотеку с открытыми исходными кодами OpenCV, использование которой не требует дополнительного дорогостоящего программного обеспечения. Использование разработанных реализаций осложняется существенным временем поиска. Реализация каскадной схемы алгоритма Latent SVM [5] позволила уменьшить время поиска объектов на тестовой базе VOC 2007 в среднем в ~18 раз. В настоящее время выполняется интеграция каскадного Latent SVM в библиотеку OpenCV. Изучается возможность повышения качества указанного алгоритма для детектирования пешеходов и транспортных средств, а также исследуются направления дальнейшей оптимизации вычислений при решении задачи многоклассового детектирования объектов.

Литература

1. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. – М.: Изд. д. Вильямс, 2004. – 465с.
2. Szeliski R. Computer Vision: Algorithms and Applications. – Springler, 2010. – 979p.
3. Sonka M., Hlavac V., Boyle R. Image Processing, Analysis and Machine Vision. – Thomson, 2008. – 866p.

4. Felzenszwalb P. F., Girshick R. B., McAllester D., Ramanan D. Object Detection with Discriminatively Trained Part Based Models // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. Vol.32, No.9. – pp. 1627–1645.
5. Felzenszwalb P. F., Girshick R. B., McAllester D., Ramanan D. Cascade object detection with deformable path model// Proceedings of the IEEE CVPR 2010.
6. Hastie T., Tibshirani R., Freidman J. The elements of statistical learning. Data mining, inference and prediction. – 2001. – 745p.
7. Viola P., Jones M.J. Robust Real-Time Face Detection // international Journal of Computer Vision 57(2). – 2004. – pp. 137-154.
8. Viola P., Jones M.J. Rapid object detection using a boosted cascade of simple features // In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition. – 2001.
9. T. Tuytelaars, M. Proesmans, L. Van Gool The cascaded Hough transform, Int'l Conf. on Image Processing, ICIP-97, vol. II, pp. 736-739, 1997.
10. Pentland A., Choudhury T. Face Recognition for Smart Environments // IEEE Computer Vision. – 2000. – pp. 50-55.
11. Alonso D., Saldaro L., Nieto M. Robust Vehicle Detection through Multidimensional Classification for On Broad Video Based Systems // IEEE. – 2007.
12. Dalal N., Triggs B. Histograms of oriented gradients for human detection // In Proceedings CVPR'05. – 2005.
13. Viola P., Jones M.J., Snow D. Detecting pedestrians using patterns of motion and appearance // In: Proceedings of the 9th International Conference on Computer Vision (ICCV), Vol. 1 – 2003. – pp. 734-741.
14. Gavrila D. M., Giebel J., Munder S. Vision-based pedestrian detection: the protector system // Proceedings of the IEEE Intelligent Vehicles Symposium, Parma, Italy. – 2004. – pp. 13-18.
15. Amit Y. 2D Object Detection and Recognition: models, algorithms and networks. – The MIT Press, 2002. – 325p.
16. Shotton J., Blake A., Cipolla R. Contour-based Learning for Object Detection // 10th IEEE International Conference on Computer Vision (ICCV'05). 2005. Vol.1. – P. 503-510.
17. Torralba A., Murphy K.P., Freeman W.T., Rubin M.A. Contex-based Vision System for Place and Object Recognition // 9th IEEE International Conference on Computer Vision (ICCV'03). 2003. Vol.1. – pp. 273-283.
18. Myung Jin Choi, Lim, J.J., Torralba, A., Willsky, A.S. Exploiting Hierarchical Contex on a large database of object categories // IEEE Computer Vision and Pattern Recognition (CVPR'10). 2010. – pp. 129-136.