Московский Государственный Университет им. М. В. Ломоносова
Механико-математический факультет
Кафедра вычислительной математики

«Допустить к защите»_____

Заведующий кафедрой

# Дипломная работа

Студента 511 группы

Ганина Ярослава Валерьевича

## «Семантическая сегментация изображений на основе метода машинного обучения»

Научный руководитель
к. ф.-м. н., с.н.с. Д. В. Иванов

Москва – 2011

**Аннотация**

В данной работе предлагается алгоритм решения следующей задачи. Каждому пикселю входного изображения некоторой сцены требуется сопоставить метку класса, так чтобы каждое связное подмножество разметки соответствовало либо одному объекту, либо группе однотипных объектов на сцене. Разработанный метод включает два этапа. На первом этапе каждое изображение тренировочного множества посредством известного алгоритма представляется в виде иерархии вложенных разбиений. В каждой такой иерархии выбираются семантически наиболее значимые фрагменты (т.е. такие, которые наилучшим образом захватывают объекты) и используются для обучения SVM классификатора. На втором этапе при помощи полученного классификатора восстанавливаются разметки новых изображений. Предлагаемый комплекс был протестирован на наборе фотографий городских улиц (The StreetScenes Challenge Framework) и превзошел по качеству оригинальный метод, основанный на фиксированных сегментациях изображений.

# Содержание

# Введение

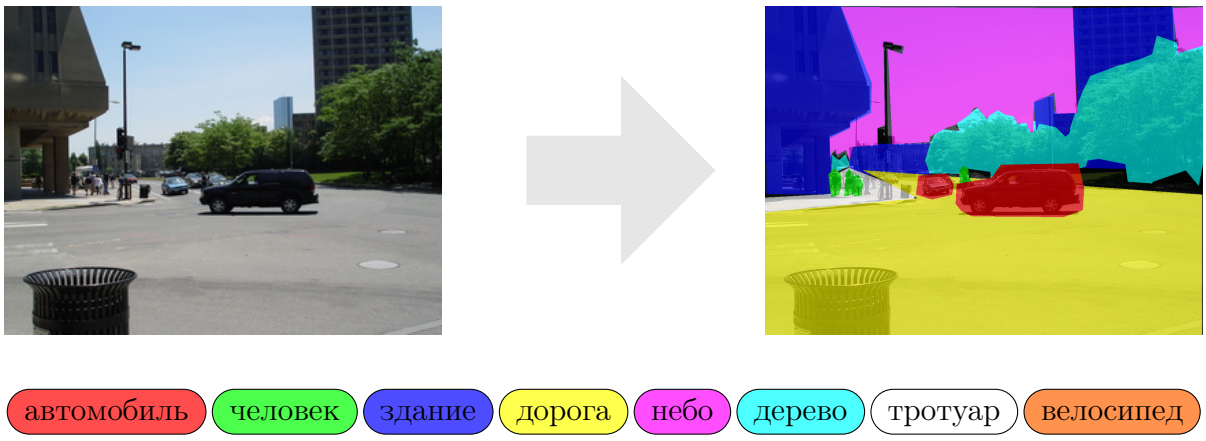



Рис. 1. Пример семантической сегментации. Входному изображению $I^{test}$ (слева) сопоставляется разметка $A^{test}$ (справа). Разные цвета регионов соответствуют разным классам объектов. См. введение.

Основной целью данной работы было разработка метода решения следующей задачи. Пусть дано множество изображений $\left\{I_i^{training}\right\}$ (например, фотографий города) и конечный набор классов объектов (например, автомобили, люди, здания), обозначаемых натуральными числами ($C = \{1, \ldots, k\}$). Пусть также каждому изображению $I_i^{training}$ сопоставлена разметка $A_i^{training}$ (т.е. кажому пикселю $I_i^{training}(x, y)$ соспоставлена метка класса $A_i^{training}(x, y) \in C$). Для входного изображения $I^{test}$ требуется восстановить такую разметку $A^{test}$, т.е. локализовать на нем все объекты заданных типов.

Рассмотрим пример. Пусть дана фотография, которая содержит людей, автомобили, несколько зданий, дорогу, растительность и небо. Необходимо разработать специальный механизм, который бы строил разбиение изображения с учетом семантики, т.е. приписывал каждому участку верную метку класса, например, как на Рис. 1. Такой процесс носит название *семантической сегментации*.

Задача разбиения изображения на отдельные содержательные части является одной из центральных для компьютерного зрения. Часто её решают при помощи построения условного Марковского случайного поля и оптимизации соотвествующей функции энергии [1–3]. Кроме локальных свойств пикселей и регионов (таких как цвет или текстура) для улучшения качества сегментации используют также глобальную или контекстную информацию (например, абсолютное [2] или относительное [4] расположение объектов разных классов, или локальные параметры соседних регионов [5]).

Подход, используемый в данной работе, основан на *машинном обучении*. Машинное обучение (МО) — это научная дисциплина, связанная с проектирование и разработкой алгоритмов, позволяющих задавать поведение ЭВМ на основе эмпирических данных, таких как показания измерительных приборов или БД. В общем случае описание всех возможных схем поведения в зависимости от поступающих данных является слишком сложной задачей, поэтому исследования в области МО фактически сосредоточены на создании механизмов автоматического описания алгоритмов.

*Обучение с учителем* — это одна из техник МО, позволяющая восстанавливать значения функций, заданной на подмножестве области определения. На этапе тренировки "ученику" предлагается набор примеров и соответвующих им значений некоторой функции. На основе полученной информации тренируемый механизм должен вывести общее правило, с помощью которого в дальшейшем принимаются решения для тестируемых данных (этап тестирования).

В данной работе используется один из методов обучения с учителем, а именно *метод опорных векторов (SVM)*. Этот довольно распространенный подход, демонстрирующий положительные результаты в ряде задач классификации [6], будет описан подробнее в следующем разделе.

Предлагаемое решение является развитием метода, описанного в кур-

совой работе [7]. Оригинальный метод включает два этапа.

На этапе тренировки для каждой входной пары $\left(I_i^{training}, A_i\right)$ строилось разбиение $I_i^{training}$ по методу, предложенному Felzenszwalb [8]. Затем каждый участок разбиения получал метку согласно $A_i$, после чего его содержимое (цвет, особые точки и т.д.) компактно описывалось вещественным многомерным вектором при помощи алгоритма извлечения характеристик $F$. Полученное множество помеченных векторов использовалось в качестве тренировочных примеров для SVM.

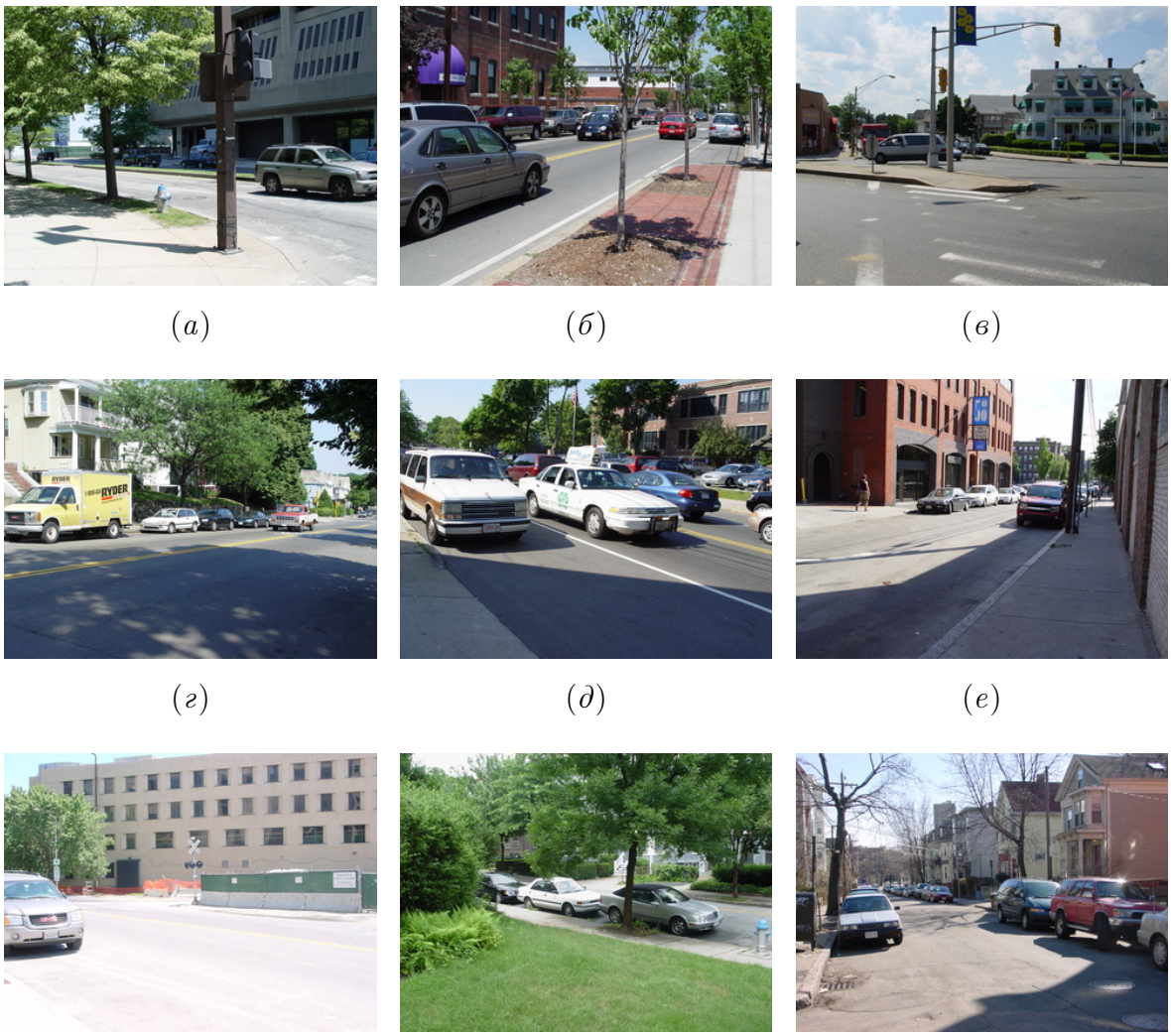
На этапе тестирования каждое входное изображение сначала сегментировалось (по тому же методу, что и на этапе тренировки). Далее при помощи F каждому сегменту сопоставлялся вектор, который подавался на вход классификатору и получал метку. Таким образом восстанавливалась разметка тестового изображения.

Одна из основных проблем такого подхода кроется в использовании алгоритма сегментации с фиксированными параметрами: малые значения параметра “хороши” для выделения небольших объектов, но приводят к дроблению более крупных; для больших значений имеет место обратный эффект. Предлагаемый в данной работе метод позволяет в некоторой степени обойти это ограничение. Для каждого изображения теперь строится специальная иерархия вложенных разбиений (*иерархическая сегментация*). Сегменты более высоких уровней иерархии получаются путем слияния наиболее “похожих” (см. раздел 1.1) регионов на низких уровнях. Тем самым увеличивается вероятность того, что каждый объект на изображении будет захвачен в “хороший” с точки зрения семантики сегмент.

Для эффективного использования новой информации о структуре изображения модифицируются процедуры тренировки классификатора и обработки тестовых изображений. Механизм восстановления разметки теперь обучается таким образом, чтобы для каждого пикселя $p$ входного изобра-

жения выбирать метку семантически “наилучшего” сегмента $S : p \in S$.

Оставшаяся часть работы организована следующим образом. Глава 1 содержит обзор используемых алгоритмов в области компьютерного зрения и машинного обучения. В главе 2 представлен метод решения поставленной задачи. Экспериментальные результаты описаны в главе 3. Итоги проделанной работы подводятся в заключении.



(*а*) (*б*) (*в*)

(*г*) (*д*) (*е*)

(*ж*) (*з*) (*и*)

Рис. 2. Примеры тренировочных изображений из базы The StreetScenes Challenge Framework [9]. См. введение и раздел 3.

# Глава 1

# Обзор используемых методов.

## 1.1. Иерархическая сегментация.

Классические алгоритмы сегментации разбивают изображение на "однородные" участки относительно низкоуровневых атрибутов (таких как цвет, яркость, текстура) по некоторым мерам сходства. Такой подход зачастую приводит к следующим проблемам:

- однородность таких атрибутов плохо согласуется с семантикой изображения;
- степень однородности участков обычно квантуется при помощи порога(ов) для данной меры.

Для определенных задач лучших результатов (по сравнению с фиксированной сегментацией) позволяет добиться *иерархическая сегментация*, которая представляет собой связанный набор разбиений изображения. Иными словами, иерархической сегментацией изображения называется дерево (или лес) фрагментов изображения (сегментов, суперпикселей), которое обладает следующими свойствами:

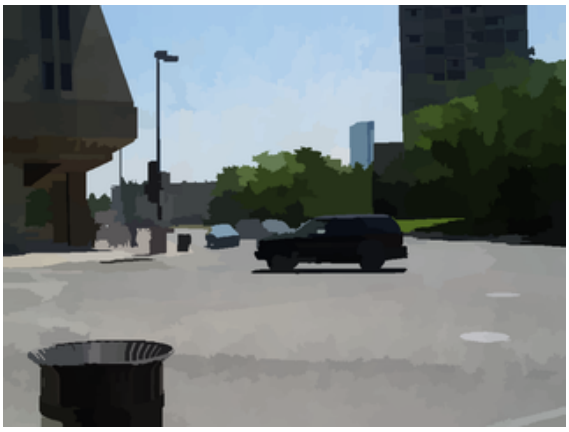
- *фрагмент-потомок* (т.е. фрагмент, соответствующий дочерней вершине в дереве; далее будем отождествлять вершины и сегменты) целиком содержится во *фрагменте-родителе*;
- объединение всех фрагментов, имеющих одинаковую глубину, дает исходное изображение.

Один из алгоритмов такой сегментации (основанный на построении минимального остовного дерева; см. [10] и [11]) будет описан далее.

(*а*). 10 уровень

(*б*). 20 уровень

(*в*). 30 уровень



(*г*). 40 уровень

(*д*). 50 уровень

Рис. 1.1. Иерархическая сегментация изображения. Примеры разбиений на различных уровнях иерархии. Сегментам соответствуют однородные по цвету фрагменты. Цвета фрагментов соответствуют средним цветам сегментов. См. раздел 1.1.

### 1.1.1. Минимальное остовное дерево весов.

Пусть $G_0(V, E, attr_v, attr_e)$ — неориентированный связный планарный граф, состоящий из конечного множества вершин $V$ и конечного множества ребер $E$ на базовом уровне пирамиды (иерархии), $attr_v : v \in V \to \mathbb{R}^+$ and $attr_e : e \in E \to \mathbb{R}^+$. Пусть каждому ребру сопоставлен неотрицательный уникальный действительный атрибут (вес). Тогда задачу сегментации можно сформулировать в терминах теории графов как поиск минимального остовного дерева для $G$ (который описывает структуру исходного изображения). Детерминистское решение этой проблемы было предложено Borûvka [12] (см. Алг. 1).

**Алг. 1** Алгоритм Borûvka

**Ввод:** Взвешенный граф $G(V, E)$.

$MST \leftarrow \{\}$
$L \leftarrow \{\}$
**for all** $v \in V$ **do**
  $L \leftarrow L \cup \{v\}$
**end for**
**while** $|L| \neq 1$ **do**
  $K \leftarrow \{\}$
  **for all** $T \in L$ **do**
    $K \leftarrow K \cup \{\arg\min (attr_e(e) | e = (u, v), u \in T, v \in G \backslash T)\}$
  **end for**
  **Merge** $L$ **using** $K$
  $MST \leftarrow MST \cup K$
**end while**

**Вывод:** Минмальное остовное дерево весов $MST$.

### 1.1.2. Иерархия разбиений.

Задача иерархической сегментации состоит в нахождении разбиения $P_k = \{CC(u_1), \ldots, CC(u_n)\}$ на $k$-м уровне пирамиды, такого что оно удовлетворяет определенным свойствам. В обсуждаемом алгоритме построение иерархии производится согласно критерию слияния регионов $Comp(CC(u_i), CC(u_j))$, который будет описан далее.

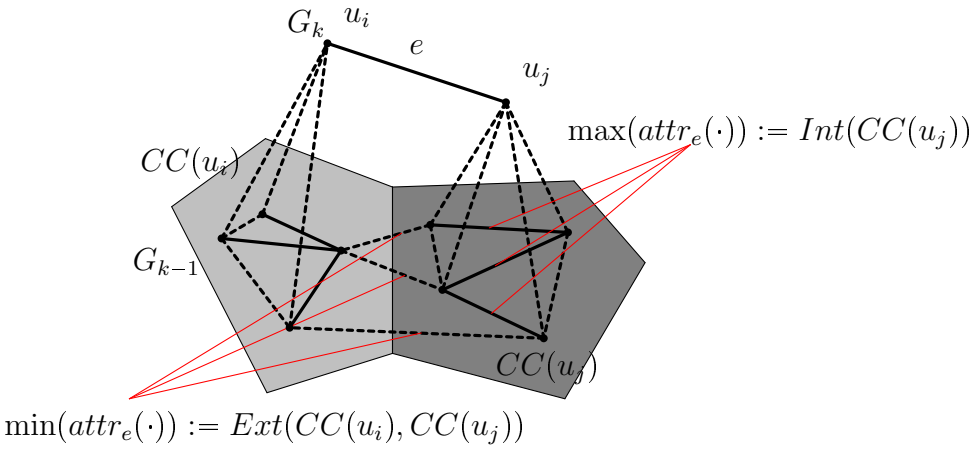



Рис. 1.2. Внутренний ($Int(\cdot)$) и внешний ($Ext(\cdot)$) контраст пары регионов. См. раздел 1.

**1. Внутренний и внешний контраст.** Пусть $G_k$ — граф на $k$-м уровне пирамиды. Каждой вершине $u_i \in G_k$ соответствует связная компонента $CC(u_i)$ разбиения $P_k$. Ядром сжатия $N_{0,k}(u_i)$ вершины $u_i \in G_k$ назовем множество сжимаемых (удаляемых) ребер базового уровня. Иными словами, применение ядра сжатия к базовому уровню сливает соответствующий подграф в $u_i$. *Внутренний контраст* компоненты $CC(u_i) \in P_k$ — это наибольшее несходство между соседними базовыми вершинами внутри $CC(u_i)$, т. е. наибольший вес ребра в $N_{0,k}(u_i)$ вершины $u_i \in G_k$ (см. Рис. 1.2):

$$Int(CC(u_i)) := \max\{attr_e, e \in N_{0,k}(u_i)\}. \tag{1.1}$$

Пусть $u_i, u_j \in V_k$ — вершины, инцидентные ребру $e \in E_k$. Назовем *внешним контрастом* между двумя компонентами $CC(u_i), CC(u_j) \in P_k$ наи-

меньший вес ребра, соединяющего $N_{0,k}(u_i)$ и $N_{0,k}(u_j)$ вершин $u_i \in CC(u_i)$ и $u_j \in CC(u_j)$ (см. Рис. 1.2):

$$Ext(CC(u_i), CC(u_j)) := \min\{attr_e(e), e = (v, w) : v \in N_{0,k}(u_i), w \in N_{0,k}(u_j)\} \tag{1.2}$$

Решающее правило для слияния двух компонент вводится как

$$Comp(CC(u_i), CC(u_j)) := \begin{cases} 1, & Ext(CC(u_i), CC(u_j)) \\ & \leqslant PInt(CC(u_i), CC(u_j)), \\ 0, & \text{иначе,} \end{cases} \tag{1.3}$$

где $PInt(CC(u_i), CC(u_j)) := \min(Int(CC(u_i)) + \tau(CC(u_i)), Int(CC(u_j)) + \tau(CC(u_j))$. Таким образом для того чтобы между двумя участками была граница (т. е. чтобы решающее правило было равно 0), внутренний контраст хотя бы одного из них должен быть больше внешнего. В формуле (1.3) в качестве порога $\tau(CC)$ может быть использована любая неотрицательная функция, определенная на множестве компонент.

**2. Построение иерархии (пирамиды) разбиений.** Алгоритм построения иерархии разбиений показан в Алг. 2. Каждая вершина $u_i \in G_k$ задает связную компоненту $CC(u_i)$ на базовом уровне пирамиды. Поскольку представленный алгоритм основан на алгоритме Borûvka, он строит минимальное остовное дерево $MST(u_i)$ каждого участка, т.е. $N_{0,k}(u_i) = MST(u_i)$. Основная идея — поиск ребер $e$ с наименьшими весами (4-й шаг), которые могли бы войти в $MST$, и сравнение $attr_e(e)$ с внутренними контрастами инцидентных ребру компонент (6-й шаг). Если вес ребра оказывается меньше, то производится слияние регионов (9-й шаг). Все такие ребра образуют ядро сжатия $N_{k,k+1}$. Иными словами, итерация алгоритма представляет собой сжатие графа $G_k$ с ядром $N_{k,k+1}$ ($G_{k+1} = C[G_k, N_{k,k+1}]$). В

**Алг. 2** Построение иерархии рабиений.

**Ввод:** Взвешенный граф $G_0$.

1: $k \leftarrow 0$

2: **repeat**

3: **for all** $u \in G_k$ **do**

4: $E_{min}(u) \leftarrow \arg\min\left(attr_e(e) | e = (u, v) \in E_k \vee e = (v, u) \in E_k\right)$

5: **end for**

6: **for all** $e = (u_i, u_j) \in E_{min} \,|\, Comp(CC(u_i), CC(u_j)) = 1$ **do**

7: $N_{k,k+1} \leftarrow N_{k,k+1} \cup e$

8: **end for**

9: $G_{k+1} \leftarrow C[G_k, N_{k,k+1}]$

10: **for all** $e_{k+1} \in G_{k+1}$ **do**

11: $attr_e(e_{k+1}) \leftarrow \min\left(attr_e(e_k) | e_{k+1} = C[e_k, N_{k,k+1}]\right)$

12: **end for**

13: $k \leftarrow k + 1$

14: **until** $G_k = G_{k+1}$

**Вывод:** Граф смежности регионов для каждого уровня пирамиды.

общем случае $N_{k,k+1}$ — лес. Для ребра $e_{k+1} \in G_{k+1}$ вес выбирается равным минимальному из весов ребер $e_k \in G_k$, сжимаемых в $e_{k+1}$ (11-й шаг). В результате на кажой итерации алгоритма строится граф смежности для областей. Каждая вершина такого графа соответсвует поддереву в минимальном остовном дереве.

## 1.2. SURF дескрипторы.

В некоторых задачах компьютерного зрения (например, в трехмерной реконструкции сцен [13]) возникает необходимость в эффективном сопо-

ставлении выделенных точек на паре изображений. Для этих целей каждая интересующая точка вместе с некоторой окрестностью описывается специальным числовым вектором (*дескриптором этой точки*), после чего между точками с "похожими" дескрипторами устанавливается соответствие.

Кроме того с помощью дескрипторов можно проводить классификацию изображений (см. [13] и разделы 1.3 и 2.1.1). В данной работе дескрипторы точек используются именно с такой целью.

Хороший дескриптор должен обладать следующими свойствами:

**Перцептуальная однородность.** Фрагменты изображения, которые выглядят "одинаковыми" для наблюдателя, должны описываться близкими (в некоторой метрике) дескрипторами. И наоборот, "различным" участкам должны соответвовать векторы, расположенные далеко друг от друга.

**Устойчивость.** Небольшие изменения следующих параметров должны слабо влиять на дескриптор:

- контраст;
- интенсивность;
- цвет;
- разрешение.

Также желательно, чтобы дескриптор был устойчив к незначительным перспективным трансформациям изображения.

**Характерность.** Различным областям соответствуют различные дескрипторы. Это означает, что дескрипторы, вычисленные для всех возможных областей, равномерно заполняют соответвующее пространство.

Один из методов построения дескрипторов точек предлагается в работе [13]. Вся процедура разбивается на два этапа.

### 1.2.1. Установление ориентации точки.

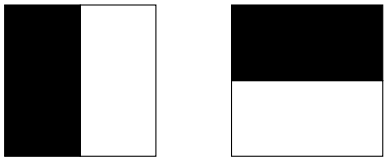
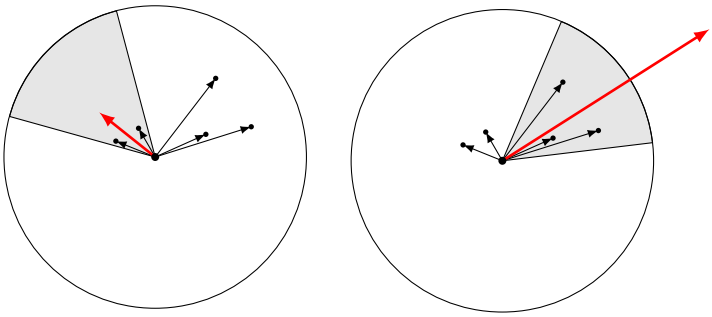


Рис. 1.3. Фильтры, вычисляющие отклики вейвлета Хаара по осям $x$ (слева) и $y$ (справа). Черным прямоугольникам соответствует вес равный -1, белым — +1. См. раздел 1.2.1.

Инвариантность дескриптора относительно вращений изображения достигается путём установления воспроизводимой ориентации точки. Для этой цели сначала вычисляются отклики вейвлета Хаара по осям $x$ и $y$ — $d_x$ и $d_y$ соответсвенно (см. Рис. 1.3) — в круговой области радиуса $6s$ вокруг интересующей точки, где $s$ — это масштаб точки. Шаг дискретизации зависит от масштаба и равен $s$. Размер вейвлетов выбирается равным $4s$.



($а$). Недоминантная ($б$). Доминантная

Рис. 1.4. Примеры ориентаций скользящих окон. См. раздел 1.2.1.

Вычисленные отклики, взвешенные при помощи фильтра Гаусса (с дисперсией $2s$) с центром в интересующей точке, представляются как радиус-векторы двухмерной плоскости с координатами $(d_x, d_y)$. Доминантная ориентация оценивается при помощи скользящего окна размера $\frac{\pi}{3}$, вращающегося вокруг начала координат. Выбирается та ориентация, которой соответствует наибольшая по евклидовой норме сумма векторов, попавших

в окно.

### 1.2.2. Описание точки при помощи суммы откликов вейвлета Хаара.

Рис. 1.5. Примеры квадратных окон (для различных масштабов), используемых для вычисления SURF дескрипторов. См. раздел 1.2.2. Источник изображения: [13].

Извлечение дескриптора начинается с конструирования квадратного региона с центром в интересующей точке и имеющего ориентацию, выбранную на предыдущем шаге. Размер региона полагается равным $20s$. Несколько примеров таких регионов изображено на Рис. 1.5.

Полученное окно разделяется на $4 \times 4$ равных квадратных подокон. Это позволяет сохранить важную пространственную информацию. Для каждого подокна в $5 \times 5$ узлах равномерной сетки вычисляются отклики вейвлета Хаара размера $2s$ по каждому направлению ($d_x$ и $d_y$) относительно выбранной ориентации (см. Рис. 1.6). Для улучшения устойчивости к геометрическим трансформациям отклики взвешиваются Гауссианом (с дисперсией $3.3s$) с центром в рассматриваемой точке.

Далее $d_x$ и $d_y$ суммируются для кажого подокна и тем самым формируют первую группу компонент искомого дескриптора. Для захвата информации о полярности изменения интенсивности также суммируются модули откликов $|d_x|$ и $|d_y|$. Таким образом для каждого подокна вычисляется четырехмерный вектор $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, который описывает

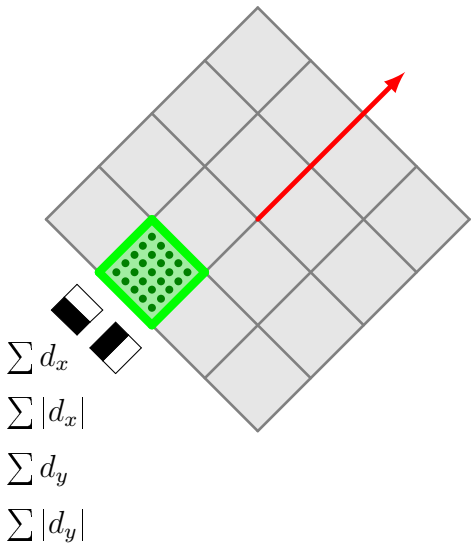



Рис. 1.6. Компоненты SURF дескриптора. Зеленый квадрат — одно из 16 подокон; темнозеленые точки обозначают узлы, в которых считаются отклики вейвлета Хаара. Отклики по обоим осям рассчитываются относительно доминантной ориентации. См. раздел 1.2.2.

внутреннюю структуру изменения интенсивности. Конкатенация $4 \times 4$ таких векторов дает 64-хмерный дескриптор. Отклики используемого вейвлета инвариантны к степени освещенности. Для достижения инвариантности к изменению контрастности полученный дескриптор нормализуется.

## 1.3. Bag-of-words.

*Гистограмма bag-of-words* — распространенный способ представления информации в задачах категоризации текстов [6, 14, 15], где он используется для описания документов при помощи частот вхождения определенных слов. Каждому слову ставится в соответствие набор тем, в которых оно употребляется наиболее часто. Учитывая такие соответствия и рассматривая гистограммы вхождения слов, специальным образом обученная система может угадывать темы предлагаемых текстов. Например, если во входном документе часто упоминаются слова “прямая”, “перпендикуляр” и

“треугольник”, то в качестве его темы будет выбрана “геометрия”.

Подобный подход в области компьютерного зрения впервые был использован Sivic и др. [16]; в работе Csurka и др. [17] показана его применимость к распознаванию объектов. Вместо текстового словаря используется словарь характеристик $D$, основанный на множестве всех характеристик тренировочных изображений (один из возможных способов составления словаря будет описан далее в разделе 2.1.1). Изображение $I$ описывается множеством характеристик $F_I$ при помощи алгоритма $F$ ($F_I = F(I)$). $F$ получает на вход изображение и строит по нему набор вещественных векторов (так называемых *дескрипторов изображения*; например, это могут быть цвета некоторых пикселей в формате $(r, g, b)$ или SURF дескрипторы особых точек). Каждой характеристике $f \in F_I$ ставится в соотвествие наиболее “похожее” нее “слово” $d_i \in D$ ($d_i = L(f)$, где $L$ — функция поиска по словарю $D$). Тогда гистограмму bag-of-words для $I$ можно задать как:

$$H_i = \sum_{f \in F_I} \mathbb{1}\left\{L(f) = d_i\right\}, \quad i = 1, \ldots, |D|. \tag{1.4}$$

Таким образом, для того чтобы описать изображение в терминах гистограммы bag-of-words необходимо задать алгоритм извлечения характеристик $F$ и функцию поиска $L$ (например, как в разделе 2.1.1)). $F$ обычно выбирают таким образом, чтобы $L$ можно было вычислить при помощи $l_2$-нормы:

$$L(f) = \arg\min_d \left[\sum_j^M (f_j - d_j)^2\right]^{\frac{1}{2}}, \quad M = |f| = |d|. \tag{1.5}$$

## 1.4. Метод опорных векторов.

*Метод опорных векторов (Support Vector Machines; SVM)* — семейство методов машинного обучения с учителем, используемых для классификации и регрессионного анализа. Бинарная (т.е. двухклассовая) задача

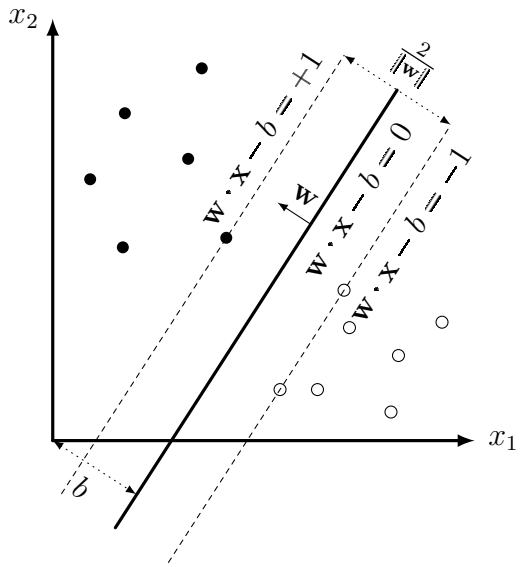



Рис. 1.7. Гиперплоскость с максимальным отступом ($\mathbf{w}\cdot\mathbf{x}-b=0$) и соотвтетствующие границы ($\mathbf{w}\cdot\mathbf{x}-b=1$ and $\mathbf{w}\cdot\mathbf{x}-b=-1$) для бинарной SVM (два класса — черные и белые точки). Точки на границах разделяющей полосы — опорные векторы. Для изображенной SVM каждая точка, попадающая в полуплоскость (относительно $\mathbf{w}\cdot\mathbf{x}-b=0$) с черными точками, классифицируется, как черная, иначе — как белая. См. раздел 1.4.

классификации, которая решается простейшим механизмом SVM, может быть описана следующим образом: для данного множества помеченных точек $\mathcal{D}=\{(\mathbf{x}_i,c_i)|\mathbf{x}_i\in\mathbb{R}^p,c_i\in\{-1,1\}\}_{i=1}^n$, где $\mathbf{x}_i$ векторы параметров, а $c_i$ — метки классов, построить правило, которое верно причисляет новую точку $\mathbf{x}$ к одному из классов.

Векторы $\mathbf{x}_i$ в данной формулировке соотвествуют объектам, а каждая компонента соответствует некоторой характеристике объекта (например, область на изображении можно характеризовать при помощи среднего цвета $(r,g,b)$, площади, периметра и т.д.).

### 1.4.1. Формализация.

Пусть даны тренировочные данные $\mathcal{D}$. Пусть каждый $\mathbf{x}_i$ — $p$-мерный действительный вектор. Требуется найти гиперплоскость, отделяющую точ-

ки с $c_i = 1$ от точек с $c_i = -1$ и притом имеющую полосу максимальной ширины (т.е. имеющую максимальное расстояние до параллельных и равноудаленных от нее гиперплоскостей) (см. Рис. 1.7). Любая гиперплоскость может быть задана набором точек $\mathbf{x}$, удовлетворяющих:

$$\mathbf{w} \cdot \mathbf{x} - b = 0\,. \tag{1.6}$$

Вектор $\mathbf{w}$ перепендикулярен плоскости. Параметр $\frac{b}{\|\mathbf{w}\|}$ определяет отступ гиперплоскости от начала координат вдоль вектора нормали $\mathbf{w}$.

$\mathbf{w}$ и $b$ выбираются из следующего соображения: необходимо максимизировать расстояние между параллельными гиперплоскостями, сохраняя их свойство быть разделяющими для тренировочных данных. Эти гиперплоскости могут быть описаны уравнениями:

$$\mathbf{w} \cdot \mathbf{x} - b = 1 \tag{1.7}$$

и

$$\mathbf{w} \cdot \mathbf{x} - b = -1. \tag{1.8}$$

Расстояние между гиперплоскостями равно $\frac{2}{\|\mathbf{w}\|}$, поэтому нужно минимизировать $\|\mathbf{w}\|$. Для того чтобы предотвратить попадание точек в разделяющую полосу, добавляется следующее условие: для каждого $i$ либо

$$\mathbf{w} \cdot \mathbf{x}_i - b \geqslant 1 \qquad \text{для } \mathbf{x}_i \text{ из первого класса,} \tag{1.9}$$

либо

$$\mathbf{w} \cdot \mathbf{x}_i - b \leqslant -1 \qquad \text{для } \mathbf{x}_i \text{ из второго класса.} \tag{1.10}$$

Это условие может быть переписано как

$$c_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geqslant 1, \qquad \text{для всех } 1 \leqslant i \leqslant n. \tag{1.11}$$

Таким образом ставится следующая задача оптимизации (по переменным $\mathbf{w}, b$):

$$\begin{cases} \|\mathbf{w}\| \to \min \\ c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geqslant 1, \quad i = 1, \ldots, n\,. \end{cases} \tag{1.12}$$

### 1.4.2. Основная форма.

Представленную выше оптимизационную задачу сложно решать по причине того, что она содержит операцию извлечения квадратного корня. Тем не менее возможно рассматривать эквивалентную задачу, в которой величина $\|\mathbf{w}\|$ заменена на $\frac{1}{2}\|\mathbf{w}\|^2$. Это задача квадратичной оптимизации (по переменным $\mathbf{w}, b$):

$$\begin{cases} \frac{1}{2}\|\mathbf{w}\|^2 \to \min \\ c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geqslant 1, \quad i = 1, \ldots, n\,. \end{cases} \tag{1.13}$$

Эта задача сводится к двойственной задаче поиска седловой точки функции Лагранжа

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\lambda}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \lambda_i \big[c_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1\big] \right\}, \tag{1.14}$$

где $\lambda_i \geqslant 0 \quad i = 1, \ldots, n$ — двойственные переменные. Заметим, что для всех точек, удовлетворяющих условию $c_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1 > 0$, соответвующие $\alpha_i$ равны нулю.

Решение задачи может быть выражено как линейная комбинация тренировочных векторов:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i c_i \mathbf{x_i}\,. \tag{1.15}$$

Как отмечалось, лишь некоторые $\alpha_i$ будут больше нуля. Соответствующие им $\mathbf{x_i}$ — это *“опорные векторы”*, лежащие на границах разделяющей

полосы и удовлетворящие $c_i(\mathbf{w}\cdot\mathbf{x_i}-b)=1$. Отсюда следует, что

$$\mathbf{w}\cdot\mathbf{x_i}-b=1/c_i=c_i \iff b=\mathbf{w}\cdot\mathbf{x_i}-c_i\,. \tag{1.16}$$

На практике сдвиг находят при помощи усреднения по всем $N_{SV}$ опорным векторам:

$$b=\frac{1}{N_{SV}}\sum_{i=1}^{N_{SV}}\left(\mathbf{w}\cdot\mathbf{x_i}-c_i\right). \tag{1.17}$$

### 1.4.3. Мягкий отступ.

В случае, когда тренировочные данные не являются линейно разделимыми (а в общем случае разделимость гарантировать не представляется возможным), используется вариант обучения с так называемым *"мягким отступом"* (предложен в 1995 г. V. Vapnik и C. Cortes [18]). Вводится набор дополнительных переменных (штрафов) $\xi_i$, характеризующих величину ошибки на $x_i$

$$c_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geqslant 1-\xi_i \quad 1\leqslant i\leqslant n\,. \tag{1.18}$$

К минимизируемому функционалу в (1.13) прибавляется функция штрафа всех ненулевых значений $\xi_i$, и задача оптимизации превращается в задачу поиска компромисса между шириной разделяющей полосы и величиной штрафа. Для линейной функции штрафа задача имеет вид:

$$\begin{cases}\left\{\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\xi_i\right\}\to\min\\ c_i(\mathbf{w}\cdot\mathbf{x_i}-b)\geqslant 1-\xi_i,\quad \xi_i\geqslant 0,\ i=1,\dots n\,.\end{cases} \tag{1.19}$$

Эта задача может быть решена при помощи метода множителей Лагранжа (как это было сделано ранее в задаче (1.13)).

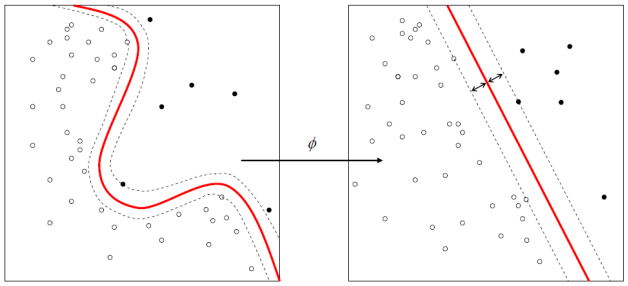



Рис. 1.8. Нелинейное преобразование исходного пространства (левый рисунок). Тренировочные данные в преобразованном пространстве могут оказаться линейно разделимыми (правый рисунок). См. раздел 1.4.4. Источник изображения: [19].

### 1.4.4. Нелинейная классификация

В 1992 г. B. Boser, I. Guyon и V. Vapnik [20] был предложен способ создания нелинейных классификаторов, в основе которого лежит переход от линейных скалярных произведений к произвольным ядрам (т.н. *kernel trick*, впервые предложенный Aizerman и др.) Такой подход позволяет оригинальному алгоритму искать разделяющую гиперплоскость в преобразованном пространстве (в котором данные могут оказаться линейно разделимыми; см Рис. 1.8). Так как размерность получаемого пространства может быть больше размерности исходного, то преобразование, сопоставляющее скалярные произведения, будет нелинейным, а значит, функция, соответствующая в исходном пространстве оптимальной разделяющей гиперплоскости, будет также нелинейной.

Если в качестве ядра взята *гауссовская радиальная базисная функция (RBF)*

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2), \quad \text{для } \gamma > 0, \tag{1.20}$$

то соответсвующее преобразованное пространство — гильбертово бесконечной размерности.

Ядро связано с преобразованием $\varphi(\mathbf{x_i})$ уравнением $k(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i}) \cdot \varphi(\mathbf{x_j})$. Гиперплоскость $\mathbf{w}$, получаемая по описанному методу, также лежит

в преобразованном пространстве ($\mathbf{w} = \sum_i \alpha_i c_i \varphi(\mathbf{x}_i)$). В процессе классификации скалярные произведения с $\mathbf{w}$ получаются по формуле: $\mathbf{w} \cdot \varphi(\mathbf{x}) = \sum_i \alpha_i c_i k(\mathbf{x}_i, \mathbf{x})$. Тем не менее в общем случае не существует такой $\mathbf{w}'$, что $\mathbf{w} \cdot \varphi(\mathbf{x}) = k(\mathbf{w}', \mathbf{x})$.

### 1.4.5. SVM для нескольких классов.

Часто бывает необходимо производить классификацию, при условии что множество меток конечно и имеет мощность $\geqslant 2$. Разработано достаточно много методов построения мультиклассовых SVM из бинарных классификаторов (например, “один-против-всех”, “один-против-одного” и др.). Другой подход, предолженный Crammer и Singer в 2001 г. [21], будет описан далее.

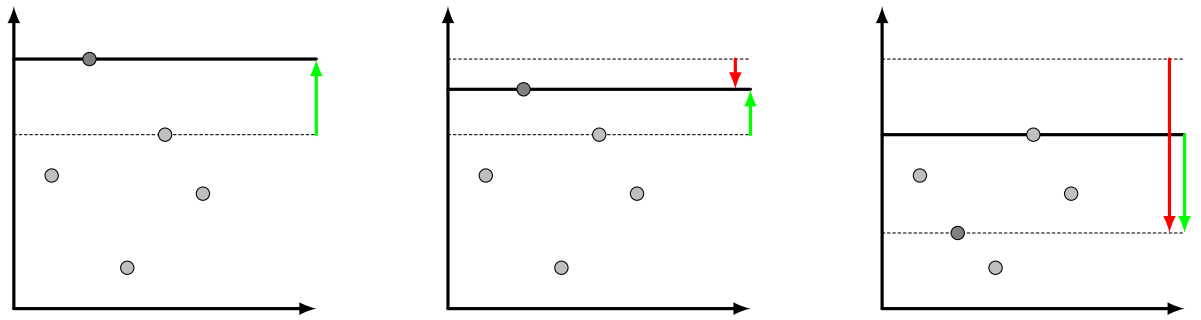
**3. CS-SVM.** Пусть дано множество из $m$ тренировочных примеров $S = \{(\mathbf{x}_i, c_i) | \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^p, c_i \in \mathcal{Y} = \{1, \ldots, k\}\}_{i=1}^m$. Тогда мультиклассовым классификатором назовем функцию $H : \mathcal{X} \to \mathcal{Y}$, которая сопоставляет элементу $\mathbf{x}$ элемент $c \in \mathcal{Y}$. Решающая функция ищется в виде:

$$H_{\mathbf{M}}(\mathbf{x}) = \arg \max_{r=1}^{k} \{M_r \cdot \mathbf{x}\} \ , \tag{1.21}$$

где $\mathbf{M}$ — вещественная матрица размера $k \times n$, а $M_r$ — $r$-я строка $\mathbf{M}$. Выражение внутри фигурных скобок назовем *уверенностью* отнесения примера $\mathbf{x}$ к классу $r$. Таким образом для $\mathbf{x}$ выбирается метка самого “уверенного” в $\mathbf{x}$ класса.

Пусть дан классификатор $H_{\mathbf{M}}(\mathbf{x})$ (параметризованный матрицей $(M)$) и пример $((x), c)$, тогда говорят, что $H_{\mathbf{M}}(\mathbf{x})$ ошибается на $(x)$, если $H_{\mathbf{M}}(\mathbf{x}) \neq c$. *Эмпирическая ошибка* для описанной задачи классификации задается формулой:

$$\epsilon_S(\mathbf{M}) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1} \left\{ H_{\mathbf{M}}((x)_i) \neq c_i \right\} . \tag{1.22}$$



(*а*) (*б*) (*в*)

Рис. 1.9. Иллюстрация функции штрафа. Темно-серая точка обозначает правильный класс. Высота каждой точки — уверенность соответствующего класса. Зеленая стрелка — отступ между верным классом и наиболее уверенным среди остальных классов. Красная стрелка — величина потери. Рис. (*а*) соответствует случаю, когда отступ $\geqslant 1$, поэтому функция штрафа равна нулю (и пример классифицирован верно). Рис. (*б*) иллюстрирует случай, когда пример классфицирован верно, но отступ недостаточно велик, и поэтому штраф не равен нулю. На рис. (*в*) изображен случай неверно классифицированного примера. См. раздел 3.

Таким образом в качестве матрицы $\mathbf{M}$ выбирается матрица, минимизирующая эмпирическую ошибку.

Для построения классификатора индикатор в формуле (1.22) заменяется кусочно линейной *функцией штрафа*

$$\max_r(M_r \cdot \mathbf{x} + 1 - \delta_{c,r}) - M_c \cdot \mathbf{x}\,, \tag{1.23}$$

где $\delta_{p,q}$ равна 1, если $p = q$, и 0 иначе. Введенная функция обнуляется, если значение уверенности правильного класса больше значений уверенности других классов как минимум на единицу. В противном случае штраф линейно пропорционален разнице между уверенностью правильного класса и максимальной уверенностью среди остальных классов (см. Рис. 1.9).

Просуммировав штрафы для всех тренировочных примеров, получаем

верхнюю границу для эмпирической ошибки:

$$\epsilon_S(\mathbf{M}) \leqslant \frac{1}{m}\sum_{i=1}^{m}\left[\max_r\{M_r\cdot\mathbf{x}_i+1-\delta_{c_i,r}\}-M_{c_i}\cdot\mathbf{x}_i\right]. \tag{1.24}$$

Тренировочная выборка $S$ линейно разделима классификатором, если существует $\mathbf{M}$, такая что штраф для всех элементов выборки равен нулю, т.е.

$$\forall i \quad \max_r\{M_r\cdot\mathbf{x}_i+1-\delta_{c_i,r}\}-M_{c_i}\cdot\mathbf{x}_i=0. \tag{1.25}$$

Отсюда следует, что для матрицы $\mathbf{M}$, удовлетворяющей (1.25), верно

$$\forall i,r \quad M_{y_i}\cdot\mathbf{x}_i+1-\delta_{c_i,r}-M_r\cdot\mathbf{x}_i\geqslant 1. \tag{1.26}$$

Введем $l_2$-норму матрицы $\mathbf{M}$ как $l_2$-норму вектора, получающегося при конкатенации строк матрицы ($\|\mathbf{M}\|_2^2=\sum_{i,j}M_{i,j}^2$)). Обобщающие свойства классификатора зависят от этой нормы, поэтому, если выборка линейно разделима, то задача сводится к поиску матрицы $\mathbf{M}$ с минимальной $l_2$-нормой, удовлетворяющей (1.26). Соответствующая задача оптимизации формулируется, как:

$$\begin{cases}\frac{1}{2}\|\mathbf{M}\|_2^2\to\min\\ \forall i,r \quad M_{y_i}\cdot\mathbf{x}_i+1-\delta_{c_i,r}-M_r\cdot\mathbf{x}_i\geqslant 1.\end{cases} \tag{1.27}$$

Заметим, что $m$ условий для $r=y_i$ выполняются автоматически. Это свойство случая линейной разделимости. В общем случае линейную разделимость, как отмечалось ранее, гарантировать нельзя. Поэтому производится переход к задаче с “мягким” отступом:

$$\begin{cases}\frac{1}{2}\|\mathbf{M}\|_2^2+C\sum\limits_{i=1}^{m}\xi_i\to\min\\ \forall i,r \quad M_{y_i}\cdot\mathbf{x}_i+1-\delta_{c_i,r}-M_r\cdot\mathbf{x}_i\geqslant 1-\xi_i.\end{cases} \tag{1.28}$$

### 1.4.6. Pegasos.

Существует несколько способов тренировки SVM. Одним из них является *метод стохастического градиентного спуска (stochastic gradient descent; SGD)*. Далее будет рассмотрена реализация этого метода на примере алгоритма Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) [22], предложенного Shalev-Schwartz, Singer и др. в 2007 г. Авторы доказывают, что количество итераций, необходимое для достижения точности $\varepsilon$ решения при данном параметре регуляризации $\lambda$, равно $O(1/\lambda\varepsilon)$ (что является существенным улучшением по сравнению с предыдущими работами по анализу SGD, где предлагается лишь $O(1/\lambda\varepsilon^2)$).

**4. Простейший алгоритм.** Пусть требуется найти бинарный классификатор $\mathbf{w}$ в задаче:

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{(\mathbf{x},c)\in S}\ell(\mathbf{w};(\mathbf{x},c)) \to \min\,, \tag{1.29}$$

где

$$\ell(\mathbf{w};(\mathbf{x},c)) = \max\{0, 1 - c(\mathbf{w}\cdot\mathbf{x})\}\ , \tag{1.30}$$

$S$ — тренировочное множество, $c \in \{-1, 1\}$. Обозначим минимизируемую функцию в (1.29) через $f(\mathbf{w})$. Тогда будем говорить, что оптимизационный метод находит *$\varepsilon$-точное решение* $\hat{\mathbf{w}}$, если $f(\hat{\mathbf{w}}) \leqslant \min_{\mathbf{w}} f(\mathbf{w}) + \varepsilon$.

Предлагаемый алгоритм нахождения $\varepsilon$-точного решения состоит в следущем. На первом шаге компоненты искомой гиперплоскости полагаются равными нулю. Далее на каждой итерации проводятся следующие действия. Пусть $t$ — номер итерации. Сначала из тренировочной выборки $S$ ($|S| = m$) случайным образом (с равномерным распределением) выбирается пример $(\mathbf{x}_{i_t}, c_{i_t})$, где $i_t \in 1, \ldots, m$. Затем минимизируемая функция в

**Алг. 3** Pegasos.

**Ввод:** $S, \lambda, T$.

$\mathbf{w}_1 \leftarrow 0$
**for** $t = 1, \ldots, T$ **do**
  $i_t \leftarrow$ **UniformRandom**$(\{1, \ldots, |S|\})$
  $\eta_t \leftarrow \frac{1}{\lambda t}$
  **if** $c_{i_t}(\mathbf{w}_t \cdot \mathbf{x}_{i_t}) < 1$ **then**
    $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t + \eta_t c_{i_t} \mathbf{x}_{i_t}$
  **else**
    $\mathbf{w}_{t+1} \leftarrow (1 - \eta_t \lambda)\mathbf{w}_t$
  **end if**
**end for**

**Вывод:** $\mathbf{w}_{T+1}$.

(1.29) заменяется на аппроксимацию:

$$f(\mathbf{w}, i_t) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(\mathbf{w}; (\mathbf{x}_{i_t}, c_{i_t}))\,. \tag{1.31}$$

Субградиент полученной функции выражается формулой

$$\nabla_t = \lambda \mathbf{w} - \mathbb{1}\left\{c_{i_t}(\mathbf{w}_t \cdot \mathbf{x}_{i_t}) < 1\right\} c_{i_t} \mathbf{x}_{i_t}\,. \tag{1.32}$$

Новое значение для разделяющей гиперплоскости вычисляется, как $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_t$, где шаг $\eta_t = 1/(\lambda t)$, что эквивалентно:

$$\mathbf{w}_{t+1} \leftarrow \left(1 - \frac{1}{t}\right)\mathbf{w}_t + \eta_t \nabla_t \mathbb{1}\left\{c_{i_t}(\mathbf{w}_t \cdot \mathbf{x}_{i_t}) < 1\right\} c_{i_t} \mathbf{x}_{i_t}\,. \tag{1.33}$$

Псевдокод алгоритма показан в Алг. 3.

**5. Проекционный шаг.** Представленный выше алгоритм можно модифицировать, добавив проекционный шаг после каждой итерации:

$$\mathbf{w}_t \leftarrow \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w_{t+1}}\|}\right\} \mathbf{w_{t+1}}\,. \tag{1.34}$$

Тем самым допустимые решения ограничиваются шаром радиуса $1/\sqrt{\lambda}$ с центром в начале координат.

# Глава 2

# Предлагаемый метод.

Разработанный алгоритм можно разбить на два этапа. Первый этап — тренировка SVM на основе набора фотографий и соотвествующих им разметок. Второй этап — восстановление разметок тестовых изображений при помощи полученного классификатора.

## 2.1. Этап тренировки.

### 2.1.1. Извлечение тренировочных дескрипторов для сегментов.

Тренировочные данные включают в себя набор сцен, содержащих объекты нескольких типов (в данной работе рассматривалось 8 классов: автомобили, люди, здания, дороги, небо, деревья, тротуары и велосипеды), а также набор соответствующих разметок (каждому объекту соспоставляется многоугольник и метка класса). Данные необходмо привести к приемлемому для классификатора виду. Поскольку алгоритм базирован на применении SVM, входные изображения и разметки преобразуются в множество многомерных вещественных векторов посредством следующей процедуры.

Входное изображение сегментируется при помощи метода, описанного в разделе 1.1. Полученная иерархия поступает на вход простейшему фильтру, который отбрасывает те сегменты, размер которых не удовлетворяет заданным ограничениям по размеру. Оставшиеся регионы описываются в терминах вещественных векторов (дескрипторов) следующим образом.

Первые 256 компонент — гистограмма *bag-of-colors (BOC)* (см. раздел 1.3), описывающая цветовое содержание фрагмента.

Оставшиеся 2048 компонент — гистограмма *bag-of-SURFs (BOS)* сег-

мента в палитре оттенков серого. Эти компоненты содержат информацию о регионе в терминах текстуры, формы, границ и т.д.

Рассмотрим подробнее процедуру расчета гистограмм. Она включает в себя два шага. Первый шаг — построение кодовых словарей. Общая схема построения словаря такова:

- каждое входное изображение помещается в равномерную сетку (для BOC узлы берутся в каждом пикселе);
- для каждого узла вычисляются дескрипторы (в случае BOS — SURF дескрипторы для 3 различных масштабов, в случае BOC — цвет);
- полученные дескрипторы помещаются в общую выборку;
- выборка кластеризуется по методу $k$-средних; центры кластеров образуют кодовый словарь.

На втором шаге при помощи полученных словарей вычисляются соответвующие гистограммы:

- для каждого сегмента ищутся содержащиеся в нем узлы сетки;
- дескриптору каждого найденного узла ставится в соответствие наиболее близкое в $\mathbb{R}^n$ (для BOC — $\mathbb{R}^3$, для BOS — $\mathbb{R}^{64}$) кодовое слово (центр кластера);
- по полученному набору кодовых слов составляется искомая гистограмма.

Таким образом иерархия регионов переводится в иерархию векторов (дескрипторов регионов) размерности $256 + 2048$.

### 2.1.2. Разметка тренировочных сегментов.

В общем случае границы разбиения, получаемого при сегментации изображения, не соотвтетствуют ручной разметке. Поэтому необходимо разработать правило отнесения каждого тренировочного сегмента к тому или иному классу. Существует несколько подходов: например, можно руководствоваться принципом минимизации симметрической разности между сегментом и областью разметки. В данной работе используется следующий метод. Пусть дана маска сегмента $M$ площади $\alpha$ ($|M| = \alpha$). Пусть $\{P_j\}$ — многоугольники, составляющие разметку изображения. Положим $M_j = M \cap P_j$, $|M_j| = \alpha_j$. Тогда в качестве метки $M$ выбирается метка многоугольника $P_{max} : \alpha_{max} \geqslant \alpha_j \ \forall j$. Однако если величина $\alpha_{max}/\alpha < \tau$, где $\tau$ — заданный порог, то сегмент исключается из тренировочной выборки.

### 2.1.3. Тренировка SVM.

**6. Препроцессинг тренировочных векторов.** Непосредственно перед тренировкой классификатора каждый вектор иерархии, описанной в разделе 2.1.1, подвергается дополнительной обработке.

В целях повышения эффективности тренируемой SVM производится покомпонентное извлечение квадратного корня. Оправданность этой операции обсуждается в [23].

Затем каждый вектор иерархии подвергается масштабированию (модифицированные гистограммы нормализуются в пространстве $l_2$). Основная цель масштабирования — предотвращение доминирования компонент вектора с более широкими числовыми границами над теми, у которых величины изменяются в достаточно узких пределах (см. “Part 2 of Sarle’s Neural Networks FAQ” [24]). Кроме того такое преобразование позволяет избежать вычислительных ошибок при замене скалярных произведений ядрами (см.

раздел 1.4.4).

**Алг. 4** ModPegasos

**Ввод:** $S, L, \lambda, T$

$\mathbf{M_1} \leftarrow 0$

**for** $t = 1, \ldots, T$ **do**

  $i_t \leftarrow$ **UniformRandom**$(\{i_{l_1}, \ldots, i_{l_{|L|}}\})$

  $B_t \leftarrow$ **GetBranch**$(i_t)$

  $\left(i_t^{(p)}, c_t^{(p)}\right) \leftarrow \arg\max_{(j,c_j)} \{|S_j| \mid S_j \triangleq \mathbf{x}_j,\ \mathbf{x}_j \in B_t\}$

  $\left(i_t^{(n)}, c_t^{(n)}\right) \leftarrow \arg\max_{(j,i)} \left\{M_{i,t} \cdot \mathbf{x}_j \mid i \neq c_t^{(p)},\ \mathbf{x}_j \in B_t\right\}$

  $\eta = \frac{1}{\lambda t}$

  $\mathbf{M}_{t+1} \leftarrow (1 - \eta\lambda) \cdot \mathbf{M}_t$

  **if** $M_{c_t^{(p)},t} \cdot \mathbf{x}_{i_t^{(p)}} < M_{c_t^{(n)},t} \cdot \mathbf{x}_{i_t^{(n)}} + 1$ **then**

    $M_{c_t^{(p)},t+1} \leftarrow M_{c_t^{(p)},t+1} + \eta \cdot \mathbf{x}_{i_t^{(p)}}$

    $M_{c_t^{(n)},t+1} \leftarrow M_{c_t^{(n)},t+1} - \eta \cdot \mathbf{x}_{i_t^{(n)}}$

  **end if**

**end for**

**Вывод:** $\mathbf{M}_{T+1}$

**7. ModPegasos.** Способ тренировки SVM классификатора, используемый в данной работе, подобен описанному в разделе 1.4.6 (кроме того см. [25]), но в отличие от него учитывает специфику задачи и структуру (иерархичность) тренировочных примеров. Опишем его пошагово (см. Алг. 4).

Пусть $S$ — тренировочное множество. Для набора изображений такое множество будет представлять собой лес, т.к. каждому изображению соответствует иерархия примеров. Пусть $L$ — множество листьев $S$ (т.е. элементов с пустым набором дочерних примеров). Будем учитывать, что алгоритм, представленный в разделе 1.1, строит вложенные разбиения, и

поэтому сегмент-потомок всегда целиком содержится в сегменте-родителе. Как и в разделе 3, необходимо найти оптимальную матрицу $\mathbf{M}$ со строками $M_i$ для задачи:

$$\frac{\lambda}{2}\|\mathbf{M}\|_2^2 + \frac{1}{|L|}\sum \ell(\mathbf{M}; (\mathbf{x}, c)) \to \min\,, \tag{2.1}$$

где $\ell(\mathbf{M}; (\mathbf{x}, c))$ — мультиклассовая функция штрафа, которая будет определена далее в тексте.

Пусть $\mathbf{M}_1 = 0$. Тогда на $t$-ой итерации алгоритма производятся следующие действия. Из $L$ случайным образом равномерно выбирается пример $l = (\mathbf{x}_{i_t}, c_{i_t})$, где $c_{i_t} \in \{1, \ldots, 8\}$ — метка класса, а $i_t \in i_{l_1}, \ldots, i_{l_{|L|}}$. Далее рассматривается ветвь дерева $B$, соответствующая $l$. В $B$ ищется пара примеров $p = (\mathbf{x}_q, c_q)$ и $n = (\mathbf{x}_r, c_s)$, которые обладают следующими свойствами. Сегмент, соответствующий $p$ — наибольший по площади среди всех сегментов ветви, а для $n$ верно:

$$(\mathbf{x}_r, c_s) = \arg\max_{(\mathbf{x}_j, i)} \left\{ M_{i,t} \cdot \mathbf{x}_j \,\middle|\, i \neq c_q \right\}\,. \tag{2.2}$$

Иными словами, $p$ — *"наилучший"* верно помеченный пример, а $n$, наоборот, — *"наихудший"* пример, на котором тренируемый классификатор ошибается. Для $p$ определяется функция штрафа:

$$\ell(\mathbf{M}_t; (\mathbf{x}_q, c_q)) = \max\left\{0, 1 + M_{c_s,t} \cdot \mathbf{x}_r - M_{c_q,t} \cdot \mathbf{x}_q\right\}\,. \tag{2.3}$$

Минимизируемая функция в (2.1) заменяется на аппроксимацию:

$$f(\mathbf{M}, i_t) = \frac{\lambda}{2}\|\mathbf{M}\|_2^2 + \ell(\mathbf{M}; (\mathbf{x}_q, c_q))\,, \tag{2.4}$$

после чего матрица $\mathbf{M}_t$ обновляется при помощи субградиента $\nabla_t$: $\mathbf{M}_{t+1} = \mathbf{M}_t - \eta_t \nabla_t$, где $\eta = 1/(\lambda t)$. Матрица субградиента $\nabla_t$ определяется, как $\nabla_t = [\nabla_{1,t}, \ldots, \nabla_{8,t}]$, где $\nabla_{i,t} = \nabla_{M_i} f(\mathbf{M}, i_t)$ — вектор-строка. Если штраф

в (2.3) равен нулю, то $\nabla_{i,t} = \lambda M_{i,t}$, иначе

$$\nabla_{i,t} = \begin{cases} \lambda M_{i,t} - \mathbf{x}_q\,, & \text{если } i = c_q; \\ \lambda M_{i,t} + \mathbf{x}_r\,, & \text{если } i = c_s; \\ \lambda M_{i,t}\,, & \text{иначе.} \end{cases} \tag{2.5}$$

Как и в оригинальном алгоритме Pegasos, обновленная матрица проектируется на замкнутое выпуклое множество $C = \left\{\mathbf{M} \,|\, \|\mathbf{M}\|_2^2 \leqslant 1/\sqrt{\lambda}\right\}$.

## 2.2. Этап тестирования.

Разметка тестового изображения $I^{test}$ восстанавливается по следующей схеме. Сначала для $I^{test}$ строится иерархия разбиений (см. раздел 1.1), которая затем преобразуется в иерархию вещественных векторов при помощи модифицированного механизма из раздела 2.1.1 со следующими изменениями:

- фильтрация сегментов не производится;
- гистограмма bag-of-words составляется согласно уже построенному в 2.1.1 кодовому словарю.

Компоненты каждого вектора подвергаются препроцессингу (см. 6).

Далее каждому пикселю $p \in I^{test}$ ставится в соответствие метка $c_p$ класса по следующему правилу:

$$c_p = \arg\max_i \left\{M_i \cdot \mathbf{x}_j \,|\, \mathbf{x}_j \in B\right\}\,, \tag{2.6}$$

где $B$ — ветвь иерархии, соответствующая $p$, а $\{M_i\}$ — строки матрицы $\mathbf{M}$ (см. раздел 7). Иными словами, $c_p$ — это метка сегмента, который содержит $p$ и при этом наиболее уверенно классифицируется SVM.

## 2.3. Замечания.

Как отмечалось выше, используемый алгоритм иерархической сегментации строит вложенные разбиения изображения. Это означает, что для любой области $P$ из заданной разметки $A$ существует единственный сегмент $S \subseteq P$, такой что $\forall S_j \subseteq P, |S| \geqslant |S_j|$. В разделе 7 такие сегменты определены как *наилучшие* для $A$.

Тренируемый классификатор таков, что на этапе тестирования, в случае если в качестве входного изображения берется тренировочное с разметкой $A$ и наилучшими сегментами $S_i$, для пикселя $p \in S_i$ он стремится выбрать метку $S_i$ и притом с нулевым штрафом (т.е. с запасом уверенности).

# Глава 3

# Результаты.

Предлагаемый метод тестировался на системе следующей конфигурации:

- **CPU:** Intel Core i7 2600K (4 ядра с тактововыми частотами 4ГГц);
- **ОЗУ:** 16Гб;
- **ОС:** Linux version 2.6.38-8-generic (buildd@allspice) (gcc version 4.5.2), Ubuntu Natty 11.04.

В качестве тренировочной базы использовалась *The StreetScenes Challenge Framework* [9]. Для обучения классификатора было выбрано 400 изображений (см. Рис. 2), 300 из которых — тренировочные данные (количество подобрано таким образом, чтобы набор рассчитанных тренировочных векторов умещался в оперативной памяти), а остальные 100 — валидационное множество (применялось для поиска оптимальных параметров классификатора). Тестирование проводилось на 100 случайных изображениях отличных от упомянутых.

## 3.1. Метод оценки качества.

Для оценки качества семантических сегментаций, производимых алгоритмом, использовалась следующая мера (*точность*):

$$\rho = \frac{\sum_{p \in P^{(test)}} \mathbb{1}\left\{c_p^{(alg)} = c_p^{(truth)}\right\}}{\left|P^{(test)}\right|}, \tag{3.1}$$

где $P^{(test)}$ — множество всех пикселей тестовых изображений, $c_p^{(alg)}$ — метка пикселя, полученная при помощи алгоритма, $c_p^{(truth)}$ — реальная метка

пикселя.

## 3.2. Результаты для метода, основанного на фиксированных разбиениях.

Результаты для нескольких конфигураций оригинального алгоритма, основанного на фиксированных сегментациях, приведены в таблице 3.1 (см. [7]). Крайний левый столбец описывает способ извлечения тренировочных сегментов (либо читаются из файла, либо генерируются алгоритмом Felzenszwalb). Второй столбец отвечает размерности гистограммы bag-of-SURFs. В третий столбец помещены ядра, используемые при тренировке SVM (см. раздел 1.4.4). Оставшаяся часть таблицы содержит значения $\rho$ при различных значениях параметра $k$ алгоритма сегментации.

Можно заметить, что прямое извлечение тренировочных регионов из ручной разметки (в качестве регионов берутся многоугольники; см. раздел 2.1.2), отрицательно сказывается на качестве семантической сегментации. Это объясняется тем, что на стадии тестирования алгоритм сегментации с очень малой долей вероятности генерирует суперпиксели, близкие к идеальным. По этой причине в данной работе такой метод извлечения не рассматривается.

Из таблицы также видно, что наилучших результатов удается добиться при использовании достаточно высокой размерности гистограмм и линейного ядра для классификатора. Поэтому предлагаемый метод базируется именно на такой комбинации (вместе с покомпонентным извлечением квадратного корня, описанным в разделе 6).

| | | | Значения параметра $k$ алгоритма Felzenszwalb | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | 500.0 | 800.0 | 1000.0 | 1300.0 | 1500.0 | 1700.0 | 2000.0 |
| Тренировочные сегменты получены из ручной разметки | 2048 | Линейное | 0.392638 | 0.453515 | 0.418384 | 0.408333 | 0.415309 | 0.475747 | 0.471167 |
| | | RBF | 0.326291 | 0.331649 | 0.311874 | 0.382448 | 0.359752 | 0.419894 | 0.424669 |
| | 1024 | Линейное | 0.372686 | 0.396029 | 0.405263 | 0.394383 | 0.39246 | 0.393165 | 0.370746 |
| | | RBF | 0.339466 | 0.346305 | 0.336582 | 0.322061 | 0.313437 | 0.306198 | 0.329108 |
| | 512 | Линейное | 0.41326 | 0.508071 | 0.475715 | 0.462077 | 0.395269 | 0.424606 | 0.44737 |
| | | RBF | 0.319144 | 0.360606 | 0.34059 | 0.377863 | 0.377691 | 0.391715 | 0.367895 |
| Тренировочные сегменты генерируются алгоритмом Felzenszwalb | 2048 | Линейное | 0.492704 | **0.678004** | 0.624416 | 0.645422 | 0.661948 | 0.674248 | 0.630718 |
| | | RBF | 0.601622 | 0.60931 | 0.626852 | 0.518661 | 0.548584 | 0.576272 | 0.500707 |
| | 1024 | Линейное | 0.621804 | 0.677689 | 0.622303 | 0.52346 | 0.632322 | 0.564095 | 0.664618 |
| | | RBF | 0.542109 | 0.636555 | 0.592278 | 0.632291 | 0.608653 | 0.555219 | 0.591704 |
| | 512 | Линейное | 0.591347 | 0.522702 | 0.64467 | 0.578042 | 0.593019 | 0.597141 | 0.651216 |
| | | RBF | 0.584374 | 0.569661 | 0.673198 | 0.616584 | 0.579944 | 0.626553 | 0.517795 |

Таблица 3.1. Результаты оригинального метода (см. [7]). Крайний левый столбец описывает метод извлечения тренировочных сегментов. Второй столбец — число кодовых слов в словаре bag-of-SURFs. Третий столбец описывает тип ядра, используемый в SVM. Оставшиеся 7 столбцов содержат величины $\rho$ для различных значений параметра алгоритма сегментации. Максимальная точность выделена жирным шрифтом. См. разделы 3.1 и 3.2.

## 3.3. Результаты для предлагаемого метода.

В таблице 3.2 представлены результаты тестирования предлагаемого в данной работе метода. Четвертая строка — количество тренировочных примеров для SVM. Значения регуляризационного коэффициента $\lambda$, используемые в ModPegasos, указаны в пятой строке.

Выбор параметра регуляризации $\lambda$ осуществлялся по следующей схеме. В [22] отмечено, что скорость сходимости итерационного процесса вычисления классификатора $\mathbf{M}$ растёт пропорционально $\lambda$. Поэтому рассматривался экспоненциально убывающий ряд $\{\lambda_i\}$ с заведомо большим значением $\lambda_1$, в котором искалось $\lambda_m$, соотвествующее наибольшей установившейся точности классификатора на валидационном наборе изображений (см. пример на Рис. 3.1). В результате $\lambda$ было положено равным $\lambda_m = 10^{-5}$.

В пятую строку таблицы помещены значения точности классификатора на валидационной выборке для последней итерации ModPegasos.

Вторая строка содержит значения точности $\rho$ для различных значений параметра $k$ алгоритма иерархической сегментации. Как и следовало ожидать $k$ практически не влияет на качество работы метода. Это связано с тем, что в процессе тренировки SVM и на стадии восстановления разметок тестовых изображений используются все уровни иерархии, а совокупность разбиений не меняется существенно при изменении константы в пороговой функции $\tau = k/|CC|$ (см. [11] и раздел 1).

На Рис. 3.2 изображены графики зависимости $\rho$ от $k$ для предлагаемого и оригинального (основанного на фиксированных сегментациях) методов. Для возможности корректного сравнения в обоих случая используются одинаковые тренировочные, тестовые и валидационные изображения. Кроме того в оригинальном методе на этапе тренировки используются векторы, вычисляемые по схеме, описанной в разделе 2.1.1. Отметим также,

что сегментация, производимая алгоритмом Felzenszwalb включена в иерархическую сегментацию при данном $k$ (см. [10]).

Хорошо видно, что эффективность оргинального метода падает с ростом $k$. Это связано с увеличением размеров элементов разбиения: в один сегмент оказываются объединены несколько достаточно крупных объектов; метка такого сегмента восстанавливается неверно, что ощутимо влияет на рассчитываемую попиксельно величину $\rho$. Однако даже при минимальных из рассматриваемых $k$ алгоритм, базирующийся на фиксированных сегментациях, проигрывает в точности локализации объектов более чем на 10%. Действительно, хотя отдельные ошибки метода на маленьких сегментах несущественны, их совокупность сильно снижает эффективность. Предлагаемый метод в некоторой степени избавлен от этих недостатков: в качестве метки каждого пикселя выбирается метка оптимального сегмента соответствующей ветви иерархии. Поэтому, как уже отмечалось, точность алгоритма на тестовом множестве изображений примерно одинакова для всех $k$.

Примеры восстановленых с помощью предлагаемого метода разметок изображены на Рис. 3.3 – 3.5. Можно заметить, что алгоритм не всегда достаточно точно захватывает объекты на сцене (а некоторые из них вообще не обнаруживает). Это может быть вызвано следующими причинами:

- *слишком грубая ручная разметка:* некоторые сегменты помимо основного объекта включают в себя значительные фрагменты окружения;
- *ограничения, связанные с методом сегментации:* хотя иерархическая сегментация позволяет добиться лучших результатов по сравнению с фиксированной, все равно не гарантирует наличие “хорошего” с семантической точки зрения сегмента для каждого объекта на сцене;
- *несбалансированность тренировочных примеров:* объекты определен-

ных классов (таких как “дорога”, “дерево”, “здание”) составляют подавляющее большинство элементов обучающей выборки, поэтому классификатор оказывается натренирован в основном именно на таких примерах (см. раздел 7);

- *недостаточно дискриминативный дескриптор для сегментов:* выбранный метод описания регионов при помощи двух гистограмм bag-of-words не учитывает, например, контекст (т.е. окружение, в котором находится сегмент);

- *наличие семантически похожих классов и неоднородных объектов:* в первую очередь идет речь о классах “дорога” и “тротуар”, которые сложно отличить и человеку, если не использовать контекст (см. выше); кроме того некоторые объекты имеют достаточно сложную структуру, и классификатор может допустить ошибку, приняв, например, окно автомобиля за окно здания.

| | Значения параметра $k$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 500 | 800.0 | 1000.0 | 1300.0 | 1500.0 | 1700.0 | 2000.0 |
| $\rho$ | 0.783005 | 0.787595 | 0.782029 | 0.779325 | 0.78742 | 0.782254 | 0.779759 |
| Количество сэмплов | 672109 | 669607 | 668745 | 668268 | 667836 | 667035 | 667025 |
| $\rho_v$ | 0.805224 | 0.807644 | 0.810811 | 0.809282 | 0.813167 | 0.797286 | 0.796023 |
| Время сборки | 2647 с | 2900 с | 2965 с | 3003 с | 3325 с | 3414 с | 3336 с |
| Время тренировки | 13253 | 13274 с | 13358 с | 13325 с | 13477 с | 13350 с | 13391 с |
| Время тестирования | 1291 | 1302 с | 1341 с | 1397 с | 1420 с | 1435 с | 1577 с |

Таблица 3.2. Результаты для предлагаемого метода. Вторая строка содержит значения параметра алгоритма иерархической сегментации. Третья строка — точность метода на тестовых примерах. Четвертая строка соответствует количеству тренировочных примеров, сгенерированных посредством сегментации. Пятая строка отвечает точности тренируемого классификатора на валидационом множестве после последней итерации ModPegasos. Оставшиеся строки содержат время, потребовавшееся для выполнения отдельных этапов метода. См. разделы 7, 3.1 и 3.3.

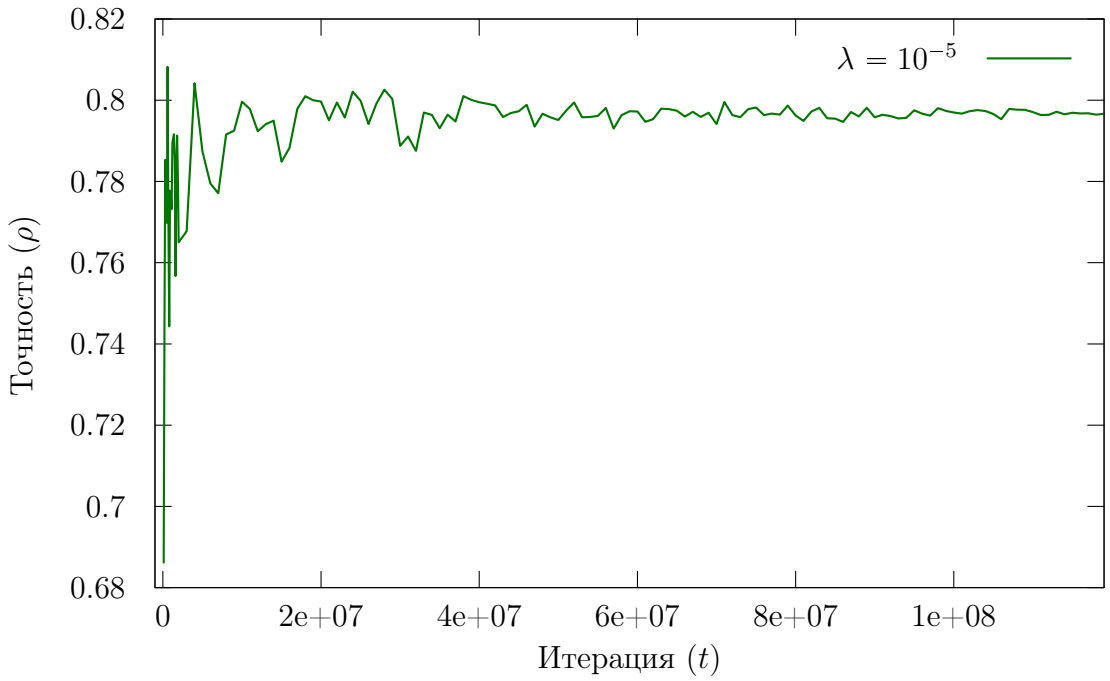



Рис. 3.1. Точность тренируемого классификатора на валидационной выборке в зависимости от номера итерации ModPegasos ($\lambda = 10^{-5}$). Ось абсцисс отвечает номеру итерации, ось ординат — величине $\rho$. См. раздел 3.3.

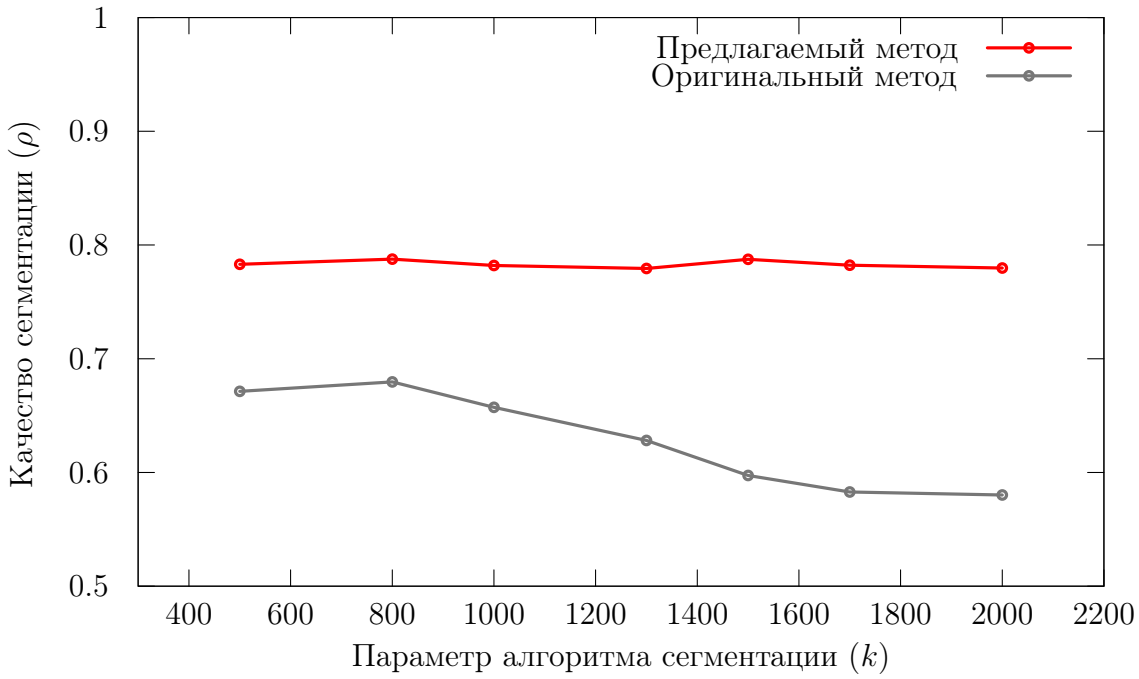



Рис. 3.2. Точности предлагаемого и оригинального методов на тестовой выборке в зависимости от параметра $k$ алгоритма сегментации. Ось абсцисс отвечает значению $k$, ось ординат – величине $\rho$. Нижний график соответствует подходу, основанному на фиксированных сегментациях. Снижение точности при росте $k$ обусловлено увеличением количества сегментов, включающих несколько крупных объектов на изображении. Верхний график соответствует предлагаемому подходу. Механизм восстановления разметки базирует свое решение на иерархии разбиений, которая не меняется существенно при различных $k$. Это объясняет, почему эффективность метода почти не зависит от параметра. См. раздел 3.

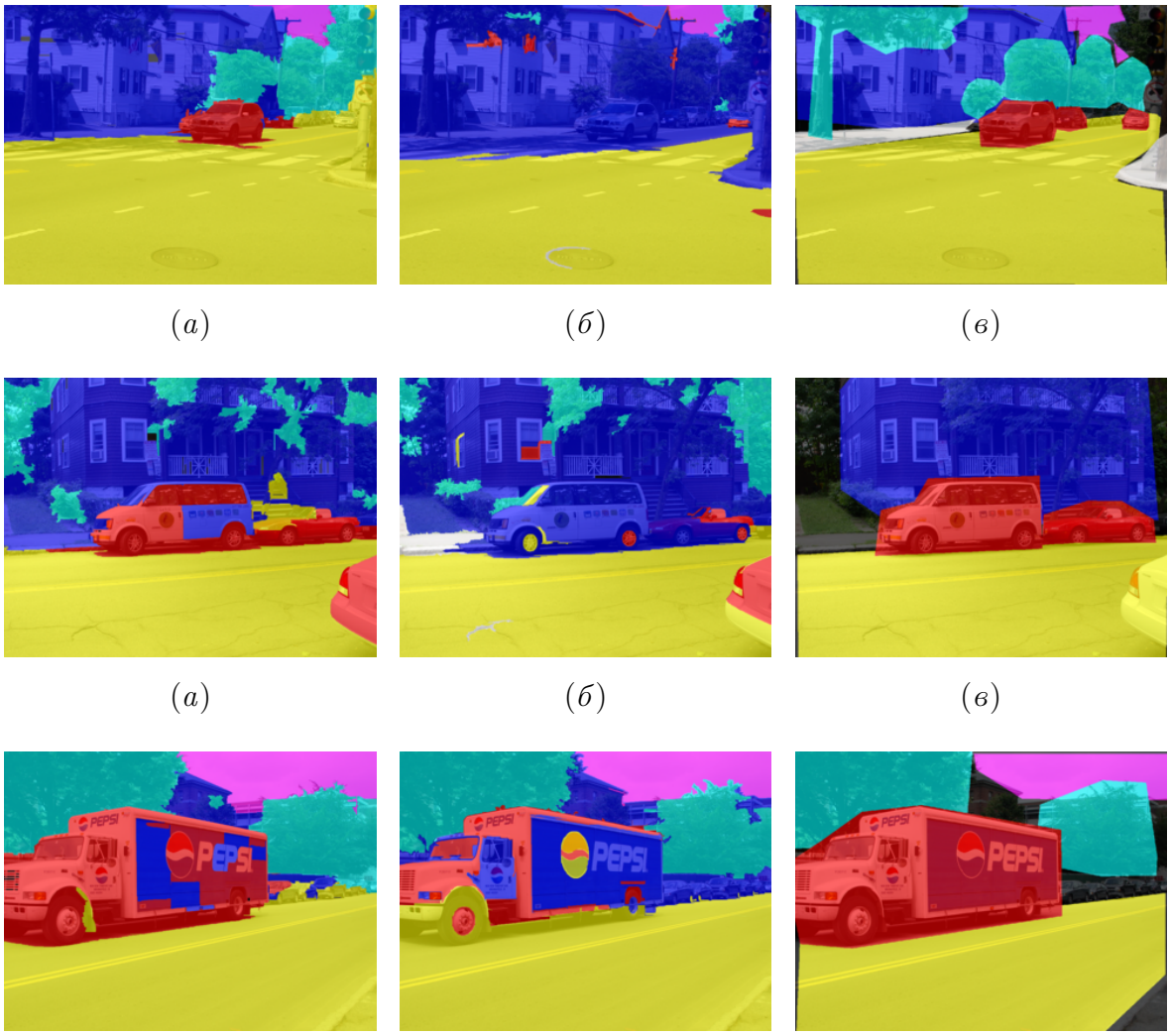



Рис. 3.3. Примеры семантических сегментаций. (*а*) — разметки, построенные предлагаемым методом; (*б*) — разметки, построенные на основе фиксированной сегментации ($k = 800$); (*в*) — ручные разметки. См. легенду на Рис. 1.

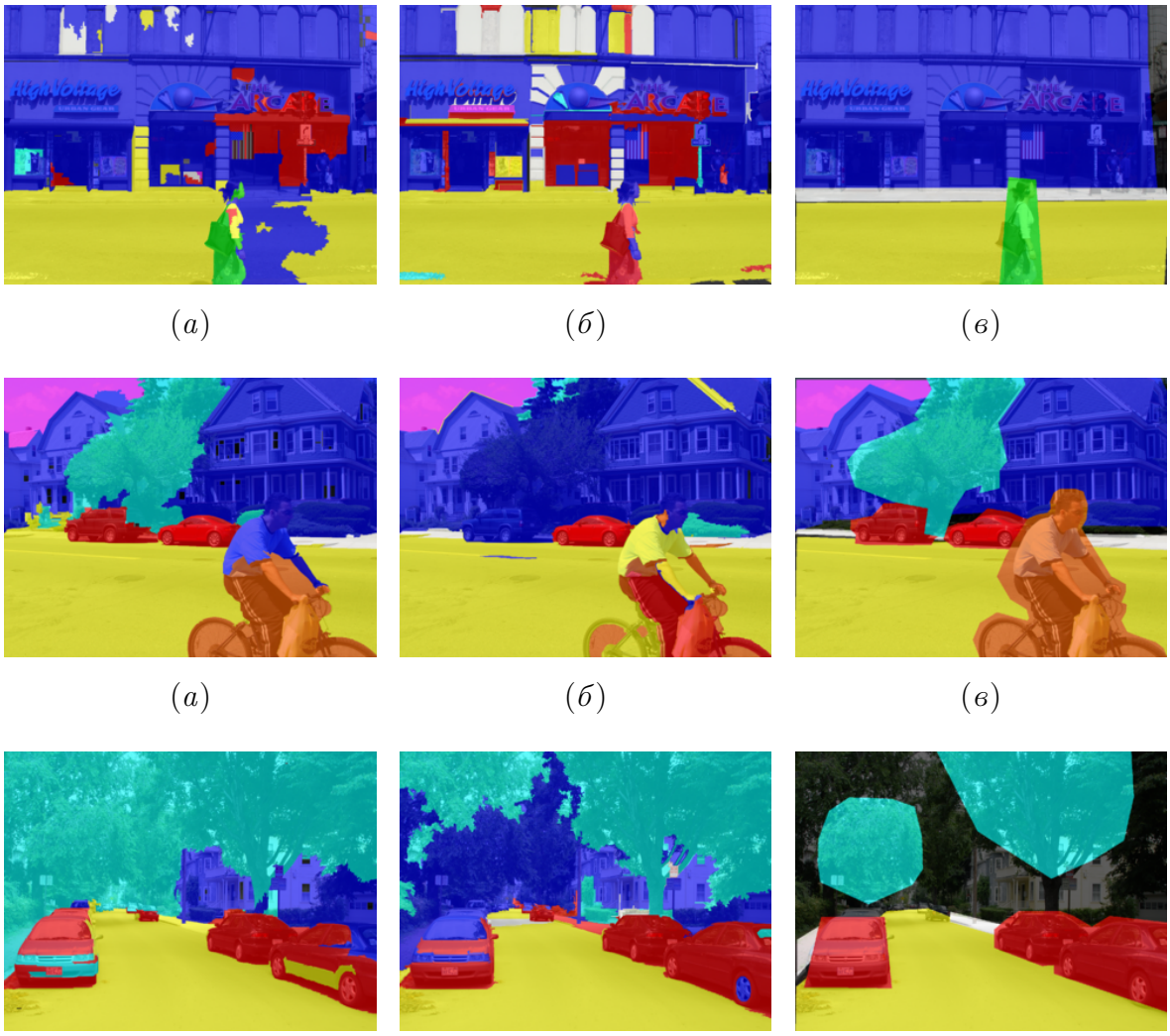
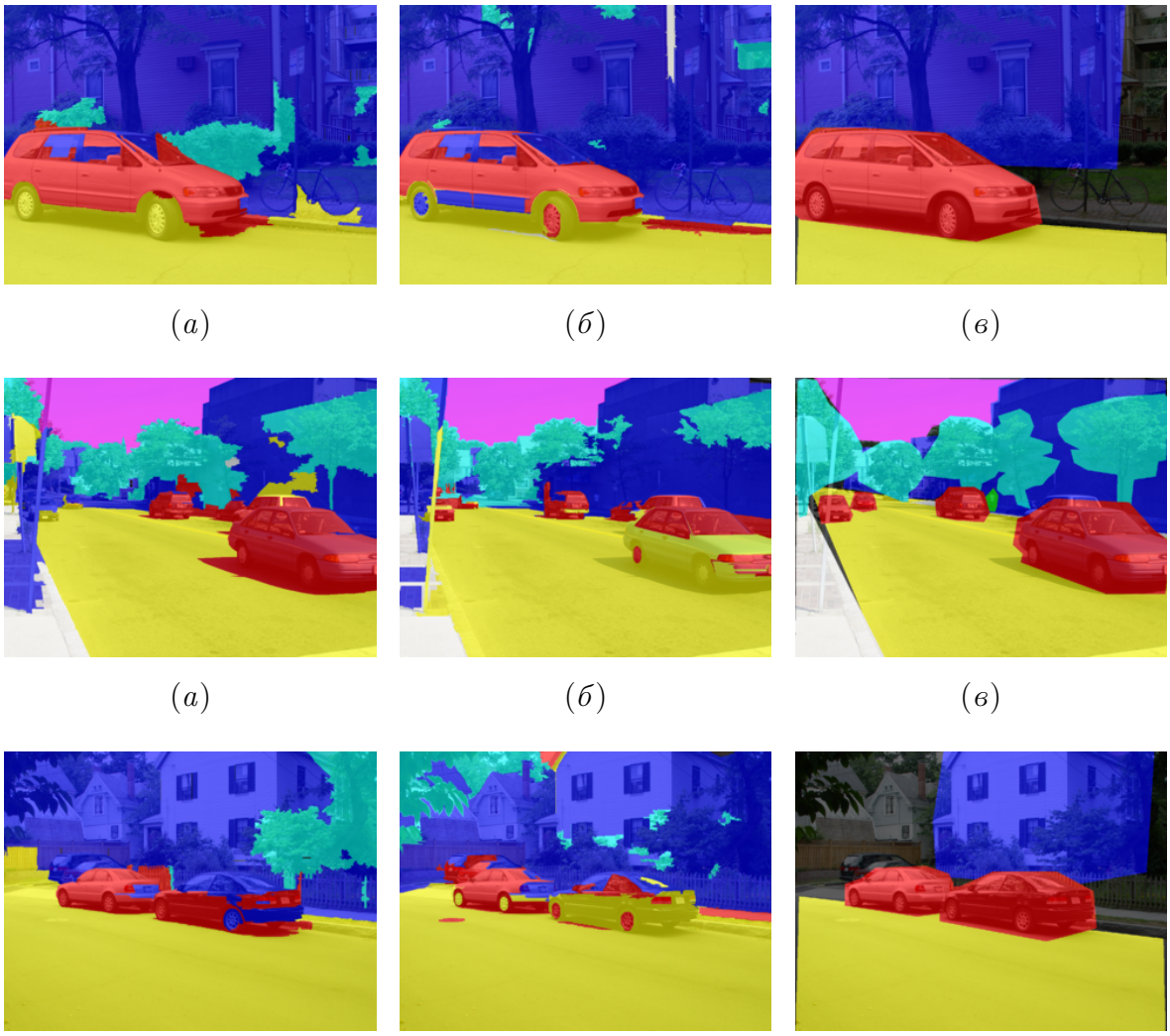



Рис. 3.4. Примеры семантических сегментаций. (*а*) — разметки, построенные предлагаемым методом; (*б*) — разметки, построенные на основе фиксированной сегментации ($k = 800$); (*в*) — ручные разметки. См. легенду на Рис. 1.



(*а*) (*б*) (*в*)

(*а*) (*б*) (*в*)

(*а*) (*б*) (*в*)

Рис. 3.5. Примеры семантических сегментаций. (*а*) — разметки, построенные предлагаемым методом; (*б*) — разметки, построенные на основе фиксированной сегментации ($k = 800$); (*в*) — ручные разметки. См. легенду на Рис. 1.

# Заключение

В данной работе был представлен метод локализации объектов заданных типов на изображениях. За основу был взят метод, описанный в [7]. Основной проблемой этого подхода являлся используемый алгоритм сегментации. Он, как и большинство подобных, не берет в расчет семантику входного изображения. В связи с этим элементы результирующего разбиения в зависимости от значения параметра часто либо являются объединением сразу нескольких объектов разных типов, либо не захватывают ни один целый объект вообще. В обоих случаях это приводит к неверной классификации: в первом — из-за того, что метка сегмента может соответствовать только одному классу, во втором — потому что для неоднородного предмета часть содержит мало информации о целом. Чтобы обойти это ограничение, в данной работе предложено использование иерархических сегментаций. Хотя в процессе построения пирамиды разбиений семантика также не учитывается, результирующая структура предоставляет возможность выбора из множества сегментов семантически наилучшего региона для каждого объекта на сцене. Как видно на Рис. 3.2, экспериментальные данные подтверждают теоретическое предположение о повышении эффективности.

Тем не менее в процессе тестирования предлагаемого метода был выявлен ряд проблем, отрицательно сказывающихся на точности восстанавливаемых разметок. Среди прочих можно выделить тот факт, что для описания регионов не используется контекст и пространственная информация. В работах [4] и [5] показано, насколько сильно учет этих характеристик может повысить эффективность подхода.

Дальнейшая модификация метода с целью устранения имеющихся недостатков, а также его экспериментальное сравнение с наиболее популярными

современными алгоритмами — предмет будущих исследований.

# Литература


1. Torralba A., Murphy K., Freeman W. Contextual models for object detection using boosted random fields / Citeseer. 2004. 6

2. Shotton J., Winn J., Rother C., Criminisi A. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation // Computer Vision–ECCV 2006. Pp. 1–15. 6

3. Kohli P., Ladickỳ L., Torr P. Robust higher order potentials for enforcing label consistency // International Journal of Computer Vision. 2009. Vol. 82, no. 3. Pp. 302–324. 6

4. Gould S., Rodgers J., Cohen D. et al. Multi-class segmentation with relative location prior // International Journal of Computer Vision. 2008. Vol. 80, no. 3. Pp. 300–316. 6, 51

5. Fulkerson B., Vedaldi A., Soatto S. Class segmentation and object localization with superpixel neighborhoods // Computer Vision, 2009 IEEE 12th International Conference on / IEEE. 2009. Pp. 670–677. 6, 51

6. Joachims T. Text categorization with support vector machines: Learning with many relevant features // Machine Learning: ECML-98. 1998. Pp. 137–142. 6, 19

7. Ganin Y. Semantic segmentation using machine learning. 2010. Coursework. 7, 40, 41, 51

8. Felzenszwalb P., Huttenlocher D. Efficient graph-based image segmentation // International Journal of Computer Vision. 2004. Vol. 59, no. 2. Pp. 167–181. 7

9. Bileschi S. StreetScenes: Towards scene understanding in still images: Ph. D. thesis / Massachusetts Institute of Technology. 2006. 9, 39

10. Haxhimusa Y., Kropatsch W. Segmentation graph hierarchies // Structural, Syntactic, and Statistical Pattern Recognition. 2004. Pp. 343–351. 10, 43

11. Ion A., Kropatsch W., Haxhimusa Y. Considerations regarding the minimum spanning tree pyramid segmentation method // Structural, Syntactic, and Statistical Pattern Recognition. 2006. Pp. 182–190. 10, 42

12. Boruvka O. On a minimal problem // Prace Moraské Pridovedecké Spolecnosti. 1926. Vol. 3. 12

13. Bay H., Tuytelaars T., Gool L. J. V. SURF: Speeded Up Robust Features // ECCV (1) / Ed. by A. Leonardis, H. Bischof, A. Pinz. Vol. 3951 of Lecture Notes in Computer Science. Springer, 2006. Pp. 404–417. 15, 16, 18

14. Cristianini N., Shawe-Taylor J., Lodhi H. Latent semantic kernels // Journal of Intelligent Information Systems. 2002. Vol. 18, no. 2. Pp. 127–152. 19

15. Lodhi H., Saunders C., Shawe-Taylor J. et al. Text classification using string kernels // The Journal of Machine Learning Research. 2002. Vol. 2. Pp. 419–444. 19

16. Sivic J., Zisserman A. Video Google: A text retrieval approach to object matching in videos. 2003. 20

17. Csurka G., Dance C., Fan L. et al. Visual categorization with bags of keypoints // Workshop on statistical learning in computer vision, ECCV / Citeseer. Vol. 1. 2004. P. 22. 20

18. Cortes C., Vapnik V. Support-Vector Networks // Machine Learning. 1995. Vol. 20, no. 3. Pp. 273–297. 24

19. Wikipedia. Support vector machine. http://en.wikipedia.org/wiki/Support_vector_machine. 25

20. Boser B. E., Guyon I., Vapnik V. A Training Algorithm for Optimal Margin Classifiers // COLT. 1992. Pp. 144–152. 25

21. Crammer K., Singer Y. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines // Journal of Machine Learning Research. 2001. Vol. 2. Pp. 265–292. 26

22. Shalev-Shwartz S., Singer Y., Srebro N. Pegasos: Primal estimated sub-gradient solver for svm // Proceedings of the 24th international conference on Machine learning / ACM. 2007. Pp. 807–814. 29, 42

23. Vedaldi A., Zisserman A. Efficient Additive Kernels via Explicit Feature Maps // IEEE Conference on Computer Vision and Pattern Recognition. 2010. 34

24. Sarle W. S. Neural Networks FAQ. 1997. http://www.faqs.org/faqs/ai-faq/neural-nets/. 34

25. Wang Z., Crammer K., Vucetic S. Multi-Class Pegasos on a Budget / Citeseer. 2010. 35