

ОГЛАВЛЕНИЕ

Предисловие	11
Требования к читателю и структура книги	12
Необходимые знания и навыки	12
Чему вы научитесь	12
Структура книги	13
Введение в машинное зрение	14
Python и NumPy	14
Обозначения и графические выделения	15
О примерах кода	16
Как с нами связаться	17
Благодарности	17
Об авторе	18
Глава 1. Основы обработки изображений	19
1.1. PIL – библиотека Python Imaging Library	19
Преобразование изображения в другой формат	20
Создание миниатюр	21
Копирование и вставка областей	21
Изменение размера и поворот	22
1.2. Библиотека Matplotlib	22
Рисование точек и прямых линий	22
Изолинии и гистограммы изображений	25
Интерактивное аннотирование	26
1.3. Пакет NumPy	27
Представление изображения в виде массива	27
Преобразование уровня яркости	29
Изменение размера изображения	31
Выравнивание гистограммы	31
Усреднение изображений	33
Метод главных компонент для изображений	34
Использование модуля pickle	37
1.4. Пакет SciPy	39
Размытие изображений	39
Производные изображений	40
Морфология – подсчет объектов	43
Полезные модули в пакете SciPy	46

1.5. Более сложный пример: очистка изображения от шумов	47
Упражнения.....	50
Соглашения в примерах кода	51

Глава 2. Локальные дескрипторы изображений 53

2.1. Детектор углов Харриса	53
Нахождение соответственных точек в изображениях	57
2.2. SIFT – масштабно-инвариантное преобразование признаков . 62	
Особые точки.....	62
Дескриптор.....	63
Обнаружение особых точек.....	63
Сопоставление дескрипторов.....	67
2.3. Сопоставление изображений с геометками	70
Загрузка изображений с геометками из Panoramio	71
Сопоставление с помощью локальных дескрипторов.....	74
Визуализация связанных изображений.....	76
Упражнения.....	78

Глава 3. Преобразования изображений 80

3.1. Гомографии	80
Алгоритм прямого линейного преобразования	82
Аффинные преобразования.....	84
3.2. Деформирование изображений.....	85
Изображение внутри изображения	86
Кусочно-аффинное деформирование	91
Регистрация изображений.....	95
3.3. Создание панорам.....	101
RANSAC	101
Устойчивое вычисление гомографии	102
Сшивка изображений.....	106
Упражнения.....	109

Глава 4. Модели камер и дополненная реальность 110

4.1. Модель камеры с точечной диафрагмой	110
Матрица камеры	111
Проецирование точек трехмерного пространства.....	113
Вычисление центра камеры	116
4.2. Калибровка камеры	116
Простой метод калибровки	117
4.3. Оценивание положения по плоскостям и маркерам	118
4.4. Дополненная реальность	123
PyGame и PyOpenGL.....	123

От матрицы камеры к формату OpenGL.....	124
Помещение виртуальных на изображение	127
Собираем все вместе	129
Загрузка моделей	130
Упражнения.....	133
Глава 5. Многовидовая геометрия	135
5.1. Эпиполярная геометрия	135
Демонстрационный набор данных	138
Построение трехмерных графиков в Matplotlib.....	140
Вычисление F – восьмиточечный алгоритм	141
Эпиполус и эпиполярные прямые	142
5.2. Вычисления, относящиеся к камерам и трехмерной структуре	145
Триангуляция	145
Вычисление матрицы камеры по точкам в пространстве.....	148
Вычисление матрицы камеры по фундаментальной матрице.....	150
5.3. Многовидовая реконструкция.....	153
Устойчивое вычисление фундаментальной матрицы.....	154
Пример трехмерной реконструкции.....	156
Обобщения и случай более двух видов.....	159
5.4. Стереои изображения	161
Вычисление карт диспаратности.....	163
Упражнения.....	167
Глава 6. Кластеризация изображений	170
6.1. Кластеризация методом K-средних	170
Пакет кластеризации в SciPy.....	171
Кластеризация изображений	172
Визуализация проекций изображений на главные компоненты	174
Кластеризация пикселей	175
6.2. Иерархическая кластеризация.....	178
Кластеризация изображений	182
6.3. Спектральная кластеризация.....	186
Упражнения.....	191
Глава 7. Поиск изображений	193
7.1. Поиск изображений по содержанию	193
Векторная модель – инструмент анализа текста	193
7.2. Визуальные слова.....	195
Создание словаря.....	195
7.3. Индексирование изображений	198
Подготовка базы данных.....	198

Добавление изображений.....	200
7.4. Поиск изображений в базе данных.....	202
Использование индекса для получения кандидатов	203
Запрос по изображению	205
Эталонное тестирование и построение графика	206
7.5. Ранжирование результатов с применением геометрических соображений.....	209
7.6. Создание демонстраций и веб-приложений	212
Создание веб-приложений с помощью CherryPy.....	212
Демонстрация поиска изображений	212
Упражнения.....	215

Глава 8. Классификация изображений по содержанию 217

8.1. Метод k-ближайших соседей.....	217
Простой двумерный пример	218
Плотные SIFT-дескрипторы в качестве признаков изображения	222
Классификация изображений – распознавание жестов	223
8.2. Байесовский классификатор	227
Использование метода главных компонент для понижения размерности.....	231
8.3. Метод опорных векторов	232
Использование библиотеки LibSVM	233
И снова о распознавании жестов	235
8.4. Оптическое распознавание символов.....	237
Обучение классификатора	238
Отбор признаков.....	238
Выделение клеток и распознавание символов	240
Выпрямление изображений	243
Упражнения.....	245

Глава 9. Сегментация изображений 247

9.1. Разрезание графов.....	247
Графы изображений	249
Сегментация с привлечением пользователя	254
9.2. Сегментация с применением кластеризации.....	258
9.3. Вариационные методы	264
Упражнения.....	265

Глава 10. OpenCV 268

10.1. Интерфейс между OpenCV и Python	268
10.2. Основы OpenCV	269

Чтение и запись изображений	269
Цветовые пространства	270
Отображение изображений и результатов обработки	270
10.3. Обработка видео	273
Ввод видео	273
Чтение видео в массивы NumPy	275
10.4. Трассировка	276
Оптический поток	276
Алгоритм Лукаса-Канаде	279
Использование трассировщика	283
Применение генераторов	284
10.5. Другие примеры	285
Ретуширование	285
Сегментация по морфологическим водоразделам	286
Обнаружение фигур с помощью преобразования Хафа	288
Упражнения	288

Приложение А. Установка пакетов 291

A.1. NumPy и SciPy	291
Windows	291
Mac OS X	291
Linux	292
A.2. Matplotlib	292
A.3. PIL	292
A.4. LibSVM	293
A.5. OpenCV	293
Windows и Unix	293
Mac OS X	294
Linux	294
A.6. VLFeat	294
A.7. PyGame	295
A.8. PyOpenGL	295
A.9. Pydot	295
A.10. Python-graph	296
A.11. Simplejson	296
A.12. PySQLite	297
A.13. CherryPy	297

Приложение Б. Наборы изображений 298

Б.1. Flickr	298
Б.2. Panoramio	299
Б.3. Оксфордская группа Visual Geometry	300

Б.4. Эталонные изображения для распознавания Кентуккийского университета	301
Б.5. Другие наборы	301
Пражский генератор данных и эталонный набор для сегментации текстур	301
Набор данных Grab Cut научно-исследовательского центра Microsoft в Кембридже	301
Caltech 101.....	302
База данных статических положений руки.....	302
Наборы стереоизображений Мидлбери-колледжа.....	302
Приложение В. Благодарности авторам изображений	303
В.1. Изображения с сайта Flickr	303
В.2. Прочие изображения.....	304
В.3. Иллюстрации	304
Литература.....	305
Предметный указатель	308



ПРЕДИСЛОВИЕ

В современном мире изображения и видео встречаются повсюду. На сайтах обмена фотографиями и в социальных сетях миллиарды таких файлов. Поисковые системы предлагают изображения едва ли не на каждый запрос. Практически все мобильные телефоны и компьютеры оснащены камерами. В телефонах многих людей хранятся гигабайты фотографий и видео.

Алгоритмы, позволяющие понять, что изображено на этих фотографиях, относятся к дисциплине, называемой «машинным зрением». Машинное зрение лежит в основе таких приложений, как поиск изображений, ориентация роботов в пространстве, анализ медицинских изображений, управление фотографиями и многих других.

Эта книга задумана как доступное введение в практические вопросы машинного зрения с изложением теоретических и алгоритмических основ и ориентирована на студентов, научных работников и энтузиастов-любителей. Для языка программирования Python имеется много отличных и при том бесплатных модулей для обработки изображений, математических вычислений и добычи данных.

Работая над этой книгой, я придерживался следующих принципов.

- Излагать материал так, чтобы у читателя возникало желание выполнять на своих компьютерах примеры по мере чтения текста.
- Использовать по преимуществу бесплатное ПО с открытым исходным кодом, не требующее много времени на начальное изучение. Python казался мне очевидным выбором.
- Полнота и замкнутость. В этой книге рассматривается не весь предмет машинного зрения, но она полна в том смысле, что весь используемый в примерах код присутствует и снабжен пояснениями. Читатель сможет повторить все примеры и на их основе писать собственные программы.
- Отдавать предпочтение широте охвата, а не деталям. Способствовать рождению идеи и побуждать к самостоятельным исследованиям, а не излагать сухую теорию.

Короче говоря, я хотел написать книгу, которая станет источником вдохновения для всех, кто интересуется программированием машинного зрения.

Требования к читателю и структура книги

В этой книге рассматриваются теоретические основы и алгоритмы решения широкого круга задач. Ниже приведено краткое описание излагаемого материала.

Необходимые знания и навыки

- Начальный опыт программирования. Вы должны знать, как пользоваться редактором и запускать скрипты, уметь структурировать код и иметь представление о базовых типах данных. Знакомство с Python или еще каким-либо скриптовым языком, например Ruby или Matlab, будет дополнительным подспорьем.
- Основы математики. Для понимания примеров следует знать о матрицах, векторах, умножении матриц, стандартных математических функциях и понятиях, в частности, о производной и градиенте. Примеры, в которых используется более сложный математический аппарат, можно пропустить без ущерба для понимания.

Чему вы научитесь

Практическое программирование обработки изображений на Python.

- Методы машинного зрения, лежащие в основе разнообразных реальных приложений.
- Многие фундаментальные алгоритмы и способы их реализации и применения в собственных программах.

В примерах демонстрируется решение следующих задач: распознавание объектов, поиск изображений по содержанию, оптическое распознавание символов, оптический поток, трассировка, трехмерная реконструкция, обработка стереоизображений, дополненная реальность, определение положения камеры, создание панорамных

изображений, очистка от шумов, группировка изображений и многое другое.

Структура книги

Глава 1 «Основы обработки изображений»

Знакомство с основными средствами работы с изображениями и используемыми модулями Python. Здесь же рассматриваются многие базовые примеры, которые потребуются в других главах.

Глава 2 «Локальные дескрипторы изображений»

Описываются методы обнаружения особых точек в изображении и использование их для поиска соответственных точек и участков в разных изображениях.

Глава 3 «Преобразования изображений»

Описываются основные преобразования изображений и методы их вычисления. Рассматриваются, в частности, деформирование изображений и создание панорам.

Глава 4 «Модели камер и дополненная реальность»

Дается введение в модели камер, создание проекций трехмерных изображений и оценивание положения камеры.

Глава 5 «Многовидовая геометрия»

Объясняется, как работать с несколькими изображениями одной и той же сцены, дается введение в многовидовую геометрию и вычисление трехмерной реконструкции изображений.

Глава 6 «Кластеризация изображений»

Описано несколько методов кластеризации и показано, как с их помощью сгруппировать изображения по сходству или по содержанию.

Глава 7 «Поиск изображений»

Показано, как хранить изображения, чтобы их можно было эффективно искать по визуальному содержанию.

Глава 8 «Классификация изображений по содержанию»

Описаны алгоритмы классификации изображений по содержанию и их применение для распознавания объектов в изображениях.

Глава 9 «Сегментация изображений»

Описаны различные методы разбиения изображений на значимые участки с применением кластеризации, интерактивного взаимодействия с пользователем или моделей изображений.

Глава 10 «OpenCV»

Показано, как использовать интерфейс из Python к популярной библиотеке машинного зрения OpenCV и как работать с данными, полученными от фото- или видеокamеры.

В конце книги приведен список литературы. Библиографические ссылки представлены номером работы в квадратных скобках, например [20].

Введение в машинное зрение

Под машинным зрением понимается автоматическое извлечение информации из изображений. В роли информации может выступать все, что угодно: 3D-модели, положение камеры, обнаружение и распознавание объектов, группировка изображений и поиск изображений по содержанию. В этой книге принято широкое определение машинного зрения, включающее такие вещи, как деформирование, очистка от шумов и дополненная реальность¹.

Иногда требуется, чтобы машинное зрение моделировало зрение человека, иногда данные подвергаются статистической обработке, а иногда ключом к решению задачи являются геометрические соображения. Мы попытаемся охватить все эти подходы.

На практике машинное зрение представляет собой сплав программирования, моделирования и математики, иногда довольно сложный для усвоения. Я сознательно стремился свести теорию к минимуму – в духе максимы "делай настолько просто, насколько возможно, но не проще". Математические пассажи необходимы для понимания алгоритмов. Некоторые главы, по самой природе материала, насыщены математикой (особенно главы 4 и 5). При желании можно пропустить всю математику и просто пользоваться кодом.

Python и NumPy

Все примеры программ в этой книге написаны на языке Python. Это лаконичный язык с хорошей поддержкой ввода-вывода, численных расчетов, обработки изображений и построения графиков. У языка есть некоторые особенности, например синтаксически значимые отступы и компактный синтаксис, к которым нужно привыкнуть. Все

¹ В этих примерах порождаются новые изображения, так что они относятся скорее к обработке изображений, чем к выделению информации как таковому.

примеры рассчитаны на версию не ниже Python 2.6, поскольку большинство пакетов доступны только для таких версий. В версиях Python 3.x имеется много отличий, не всегда совместимых с Python 2.x и с экосистемой нужных нам пакетов (но это временно).

Знакомство с основами Python облегчит понимание материала. Для начинающих хорошими отправными точками могут стать книга Mark Lutz «Learning Python» [20] и документация на сайте <http://www.python.org/>.

При программировании машинного зрения необходимы средства для представления векторов и матриц и операций над ними. Все это есть в модуле NumPy, где векторы и матрицы представлены типом `array`. Такое же представление используется и для изображений. Хорошим справочным пособием по модулю NumPy является бесплатная книга Travis Oliphant «Guide to NumPy» [24]. Тем, кто только начинает осваивать NumPy, поможет также документация на сайте <http://num-py.scipy.org/>. Для визуализации результатов мы пользуемся модулем `matplotlib`, а для более сложных математических вычислений — модулем `SciPy`. Это основные пакеты, знакомству с ними посвящена глава 1.

Помимо основных, есть много других бесплатных пакетов на Python, применяемых для таких задач, как чтение данных в формате JSON и XML, загрузка и сохранение данных, генерация графиков, графическое программирование, создание демонстраций в вебе, построение классификаторов и многих других. Обычно они нужны только для разработки определенных приложений или демонстраций, в противном случае их можно не устанавливать.

Заслуживает упоминания также интерактивная оболочка IPython, упрощающая отладку и экспериментирование. Скачать ее (вместе с документацией) можно с сайта <http://ipython.org/>.

Обозначения и графические выделения

Код выглядит следующим образом:

```
# точки
x = [100,100,400,400]
y = [200,500,200,500]

# нанести точки на график
plot(x,y)
```

В книге применяются следующие графические выделения.

Курсив

Определения терминов, имена файлов и переменных.

Моноширинный

Имена функций, модулей Python, примеры кода и вывод на консоль.

Гиперссылка

URL-адреса

Простой текст

Все остальное.

Математические формулы приводятся либо в основном тексте, например $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$, либо в отдельной строке:

$$f(\mathbf{x}) = \sum_i w_i x_i + b$$

Формулы нумеруются, только если на них имеются ссылки.

Скалярные величины обозначаются строчными буквами ($s, r, \lambda, \theta, \dots$), матрицы – заглавными буквами (A, V, H, \dots) (буквой I обозначается изображение, представленное в виде массива), а векторы – полужирными строчными буквами ($\mathbf{t}, \mathbf{c}, \dots$). Точки на плоскости (в изображении) и трехмерном пространстве обозначаются соответственно $\mathbf{x} = [x, y]$ и $\mathbf{X} = [X, Y, Z]$.

О примерах кода

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешение необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, никто не возражает включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров из книг издательства O'Reilly на компакт-диске разрешение требуется. Цитировать книгу и примеры в ответах на вопросы можно без ограничений. Но для включения значительных объемов кода в документацию по собственному продукту нужно получить разрешение.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: Jan Erik Solem «Programming Computer Vision

with Python» (O'Reilly). Copyright © 2012 Jan Erik Solem, 978-1-449-31654-9.

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу permissions@oreilly.com.

Как с нами связаться

Вопросы и замечания по поводу этой книги отправляйте в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
707-829-0515 (международный или местный)
707-829-0104 (факс)

Для этой книги имеется веб-страница, на которой публикуются списки замеченных ошибок, примеры и прочая дополнительная информация. Адрес страницы: http://oreil.ly/comp_vision_w_python.

Замечания и вопросы технического характера следует отправлять по адресу bookquestions@oreilly.com.

Дополнительную информацию о наших книгах, конференциях и новостях вы можете найти на нашем сайте по адресу <http://www.oreilly.com>.

Читайте нас на Facebook: <http://facebook.com/oreilly>.

Следите за нашей лентой в Twitter: <http://twitter.com/oreillymedia>.

Смотрите нас на YouTube: <http://www.youtube.com/oreillymedia>.

Благодарности

Я выражаю благодарность всем участвовавшим в процессе подготовки и производства книги. Мне помогли многие сотрудники издательства O'Reilly. Отдельное спасибо редактору Энди Ораму (O'Reilly) и Полу Анагностопулосу (Windfall Software), помогавшему в процессе производства.

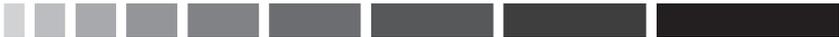
Многие высказывали замечания по поводу различных черновых редакций книги, которые я выкладывал в сеть. Особо хочу отметить заслуги Класа Джозефсона (Klas Josephson) и Хакана Ардё (Håkan Ardö), приславших подробные замечания и отзывы. Фредрик Кал (Fredrik Kahl) и Пау Гаргалло (Pau Gargallo) помогли в проверке

фактов. Спасибо всем читателям за слова ободрения и за стремление улучшить текст и код. Письма от незнакомых людей, высказывающих свои мысли о черновике рукописи, – мощный источник мотивации.

Напоследок я хочу поблагодарить своих друзей и членов семьи за поддержку и понимание на протяжении всего времени, когда я посвящал вечера и выходные писанию книги. А особенно свою жену, Сару, которая давно уже поддерживает меня во всем.

Об авторе

Ян Эрик Солем, большой энтузиаст языка Python, занимается исследованиями и популяризацией машинного зрения. По образованию прикладной математик, работал доцентом, техническим директором стартапа, пишет книги. Иногда рассказывает о машинном зрении и Python в своем блоге по адресу www.janeriksolem.net. Много лет использует Python для преподавания, исследований и разработки промышленных приложений в области машинного зрения. В настоящее время проживает в Сан-Франциско.



ГЛАВА 1.

Основы обработки изображений

Эта глава представляет собой введение в обработку изображений. На целом ряде примеров иллюстрируется использование основных пакетов Python, применяемых для работы с изображениями. Здесь мы познакомимся с базовыми средствами чтения изображений, их преобразования и масштабирования, вычисления производных, сохранения результатов, построения графиков и т. д. Все это понадобится в последующих главах.

1.1. PIL – библиотека Python Imaging Library

Библиотека *Python Imaging Library (PIL)* содержит общие средства для обработки изображений и разнообразных полезных операций, в том числе: изменение размера, кадрирование, поворот, преобразование цветов и т. д. Библиотека распространяется бесплатно, ее можно скачать с сайта <http://www.pythonware.com/products/pil/>.

PIL позволяет читать изображения, записанные в большинстве существующих форматов, и сохранять в наиболее популярных. Наиболее важен модуль `Image`. Для чтения изображения служит такой код:

```
from PIL import Image

pil_im = Image.open('empire.jpg')
```

В качестве значения метод возвращает объект изображения *pil_im*. Для преобразования цветов используется метод `convert()`. Чтобы прочитать изображение и сделать его полутоновым, достаточно просто добавить вызов `convert('L')`:

```
pil_im = Image.open('empire.jpg').convert('L')
```

Ниже приведено несколько примеров, заимствованных из документации по PIL на странице <http://www.pythonware.com/library/pil/handbook/index.htm>. Результаты их работы показаны на рис. 1.1.



Рис. 1.1. Результаты обработки изображений с помощью PIL

Преобразование изображения в другой формат

С помощью метода `save()` PIL может сохранять изображения в большинстве графических форматов. В следующем примере мы читаем изображения из файлов, перечисленных в списке `filelist`, и преобразуем их в формат JPEG:

```
from PIL import Image
import os

for infile in filelist:
    outfile = os.path.splitext(infile)[0] + ".jpg"
    if infile != outfile:
        try:
            Image.open(infile).save(outfile)
        except IOError:
            print "не могу преобразовать", infile
```

Функция `open()` создает объект изображения PIL, а метод `save()` сохраняет это изображение в файле с указанным именем. Новое имя файла совпадает с исходным за исключением расширения, которое теперь равно «.jpg». PIL умеет определять формат файла по расширению имени. Проверяется, что исходный формат файла отличен от JPEG, а в случае ошибки преобразования печатается сообщение.

В этой книге мы не раз будем задавать списки файлов, подлежащих обработке. Вот, например, как можно сформировать список имен всех графических файлов в папке. Создайте файл *imtools.py*, в который мы будем помещать полезные функции общего назначения, и добавьте в него такую функцию:

```
import os

def get_imlist(path):
    """ Возвращает список имен всех
        jpg-файлов в каталоге. """
    return [os.path.join(path, f) for f in os.listdir(path)
            if f.endswith('.jpg')]
```

А теперь вернемся к PIL.

Создание миниатюр

Создавать миниатюры с помощью PIL очень просто. Метод `thumbnail()` принимает кортеж, в котором задается новый размер, и преобразует изображение в миниатюру указанного размера. Вот как создается миниатюра, для которой длина наибольшей стороны равна 128 пикселей:

```
pil_im.thumbnail((128,128))
```

Копирование и вставка областей

Обрезка (кадрирование) изображения производится методом `crop()`:

```
box = (100,100,400,400)
region = pil_im.crop(box)
```

Прямоугольная область определяется 4-кортежем, в котором задаются координаты сторон в порядке (левая, верхняя, правая, нижняя). В PIL используется система координат с началом (0, 0) в левом верхнем углу. С вырезанной областью можно затем производить различные операции, например повернуть и вставить в то же место методом `paste()`:

```
region = region.transpose(Image.ROTATE_180)
pil_im.paste(region,box)
```

Изменение размера и поворот

Для изменения размера изображения служит метод `resize()`, которому передается кортеж, определяющий новый размер:

```
out = pil_im.resize((128,128))
```

Для поворота изображения вызывается метод `rotate()` и задается угол в направлении против часовой стрелки:

```
out = pil_im.rotate(45)
```

Результаты работы некоторых примеров показаны на рис. 1.1. Слева показано исходное изображение, затем его полутоновый вариант, результат вставки вырезанной и повернутой области и, наконец, миниатюра.

1.2. Библиотека Matplotlib

Для построения графиков, а также рисования точек и прямых или кривых линий на изображении применяется графическая библиотека `matplotlib`, содержащая куда больше средств построения графиков, чем имеется в `PIL`. `matplotlib` генерирует высококачественные рисунки, с ее помощью получены многие иллюстрации к этой книге. Интерфейс `PyLab`, включенный в `matplotlib`, — это набор функций, позволяющих пользователю строить графики. Исходный текст `matplotlib` открыт, скачать библиотеку можно бесплатно с сайта <http://matplotlib.sourceforge.net/>, где имеется также подробная документация и учебные пособия. Ниже приведено несколько примеров использования функций, которые понадобятся в этой книге.

Рисование точек и прямых линий

Хотя библиотека позволяет создавать красивые столбчатые и секторные диаграммы, диаграммы рассеяния и т. п., для целей машинного зрения достаточно всего нескольких команд. Нам нужно выделять особые точки, соответственные области и обнаруженные объекты с помощью точек и прямых линий. Вот пример нанесения на изображение нескольких точек и отрезка прямой:

```
from PIL import Image
```

```
from pylab import *

# прочитать изображение в массив
im = array(Image.open('empire.jpg'))

# поместить на график изображение
imshow(im)

# несколько точек
x = [100,100,400,400]
y = [200,500,200,500]

# нанести точки в виде красных звездочек
plot(x,y,'r*')

# нарисовать отрезок, соединяющий первые две точки
plot(x[:2],y[:2])

# добавить заголовок и показать график
title('Plotting: "empire.jpg"')
show()
```

Здесь на график помещается изображение, затем четыре точки, обозначенные красными звездочками (их координаты задаются в списках x и y), и, наконец, отрезок прямой, соединяющий первые две точки из списка (по умолчанию рисуется синим цветом). Результат показан на рис. 1.2. Функция `show()` открывает графический интерфейс рисунка и создает окна. Цикл обработки сообщений в ГИП блокирует выполнение скриптов на все время, пока не будет закрыто последнее окно рисунка. Вызывать `show()` следует только один раз, обычно в конце скрипта. Обратите внимание, что в PyLab начало координат расположено в левом верхнем углу, как принято при работе с изображениями. Оси полезны для отладки, но если хотите получить более красивую картинку, то можете отключить их:

```
axis('off')
```

Тогда график будет выглядеть, как на рис. 1.2 справа.

Существует много параметров для задания цвета и стиля. Самые полезные короткие команды приведены в таблицах 1.1, 1.2 и 1.3. Используются они так:

```
plot(x,y)           # по умолчанию сплошная синяя линия
plot(x,y,'r*')      # красные маркеры в виде звездочек
plot(x,y,'go-')     # зеленая линия с маркерами-кружочками
plot(x,y,'ks:')     # черная пунктирная линия с маркерами-квадратиками
```

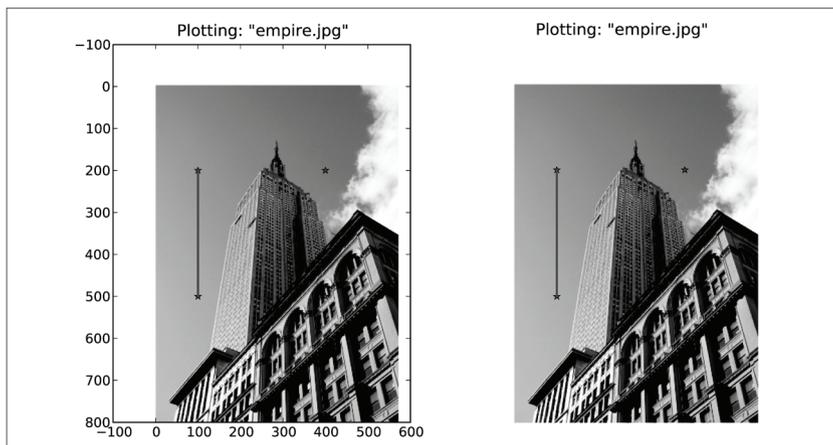


Рис. 1.2. Примеры графиков, построенных с помощью `Matplotlib`.
Изображение с точками и отрезком прямой
с осями координат и без них

Таблица 1.1. Основные команды задания цветов
при построении графиков в `PyLab`

Цвет

'b'	синий
'g'	зеленый
'r'	красный
'c'	голубой
'm'	пурпурный
'y'	желтый
'k'	черный
'w'	белый

Таблица 1.2. Основные команды задания цветов
при построении графиков в `PyLab`

Line style

'-'	solid
'--'	dashed
':'	dotted

Таблица 1.3. Основные команды задания маркеров при построении графиков в PyLab

Маркер	
'.'	точка
'o'	кружочек
's'	квадратик
'*'	звездочка
'+'	плюс
'x'	x

Изолинии и гистограммы изображений

Рассмотрим два примера специальных графиков: изолинии и гистограммы изображений. Визуализация изолиний изображения (или изолиний других двумерных функций) может оказаться очень полезной. Для этого нужны полутоновые изображения, потому что изолинии строятся по значению какой-нибудь одной величины. Делается это так:

```
from PIL import Image
from pylab import *

# прочитать изображение в массив
im = array(Image.open('empire.jpg').convert('L'))

# создать новый рисунок
figure()

# не использовать цвета
gray()

# показать изолинии относительно левого верхнего угла
contour(im, origin='image')
axis('equal')
axis('off')
```

Как и раньше, метод `convert()` преобразует исходное изображение в полутоновое.

Гистограмма изображения – это график распределения значений пикселей. Область возможных значений разбивается на интервалы, и для каждого интервала определяется количество пикселей, значения которых попадают в этот интервал. Для построения гистограммы полутонового изображения применяется функция `hist()`:

```
figure()
hist(im.flatten(), 128)
show()
```

Ее второй аргумент задает количество интервалов. Отметим, что изображение сначала необходимо линейризовать, потому что `hist()` ожидает получить одномерный массив. Метод `flatten()` преобразует любой массив в одномерный, располагая значения по строкам. На рис. 1.3 показаны изолинии и гистограмма.

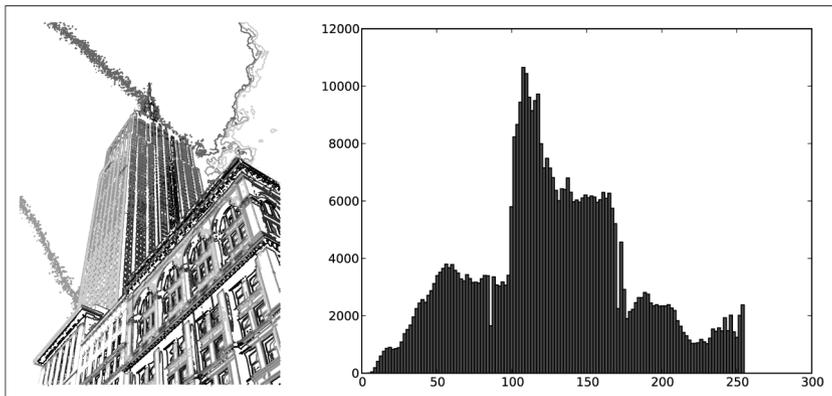


Рис. 1.3. Examples of visualizing image contours and plotting image histograms with Matplotlib

Интерактивное аннотирование

Иногда пользователям нужно взаимодействовать с приложением, например пометить какие-то точки изображения или аннотировать обучающие данные. В PyLab для этого имеется простая функция `ginput()`:

```
from PIL import Image
from pylab import *

im = array(Image.open('empire.jpg'))
imshow(im)
print 'Щелкните в 3 точках'
x = ginput(3)
print 'вы щелкнули:', x
show()
```

После вывода графика эта программа будет ждать, пока пользователь три раза щелкнет мышью в области окна рисунка. Координаты $[x, y]$ отмеченных точек сохраняются в списке `x`.