Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Курсовая работа на тему:

«Детектор элементов лица для построения псевдотрехмерной графики»

Лебедев Дмитрий, 361 гр.

Научный руководитель:

Пименов Александр

Санкт-Петербург

2011

Введение

В последнее время стала очень популярна активно развивающаяся технология 3D. Она используется во многих областях, начиная от проектирования различных объектов, заканчивая производством фильмов и 3D телевизоров. Преимущество трехмерного представления данных перед двумерным очевидно, так как достигается большая наглядность. С появлением высококачественного видеооборудования по доступной цене стали актуальны такие задачи как «восстановление трехмерной структуры объекта по видеоряду», «выделение и отслеживание объектов из видео-потока», «распознавание жестов и объектов» и многие другие.

В этой работе рассматривается группа алгоритмов адаптивного бустинга AdaBoost предложенная Yoav Freund и Robert E. Schapire для построения сильного классификатора из нескольких примитивных, метод классификации лиц на основе примитивов Haar'а и техника построения каскада классификаторов, предложенная Paul Viola и Michael Jones.

Целью этой работы является построение детектора глаз человека на основе алгоритма AdaBoost, необходимого для решения следующей более емкой задачи – проектирования псевдотрехмерного изображения с помощью полученного детектора.

Постановка задачи

Целью данной работы:

- изучить алгоритм AdaBoost, технику построения каскада классификаторов Viola-Jones
- изучить библиотеку OpenCV[1]
- собрать данные, необходимые для тренировки каскада
- построить каскад классификаторов, с помощью библиотеки OpenCV

AdaBoost

Boosting – процесс улучшения.

AdaBoost – семейство алгоритмов предложенное Freund и Schapire.
На входе алгоритму подается множество пар S = { (x_1, y_1), …, (x_n, y_n) } в которых элемент x_i принадлежит некому множеству X, а элемент y_i принадлежит множеству Y. Для лучшего понимания можно представлять элементы множества Y как пометки присвоенные элементам множества X. Мы рассмотрим только случай когда Y – двухэлементное множество, то есть Y = {-1, +1}. В общем случае h_t имеет вид h_t : X -> Y, но так как в нашем случае Y – двухэлементное множество, то h_t примет вид h_t : X -> {-1, +1}. На каждом шаге алгоритм вызывает классификаторы $h_j \in H$ и подает на вход пары из множества S, называемого обучающей выборкой, с распределенными весами w_t(i), значения которых будут означать цену ошибки. Изначально веса каждого примера равны, но на каждом шаге вес некорректно определенного примера возрастает. Идея заключается в том, что слабые классификаторы, вызываясь повторно на разных этапах, будут концентрироваться на тех примерах, которые имеют больший вес. Алгоритм является адаптивным, в том смысле, что при обучении он обращает внимание на те примеры, которые были неверно классифицированы.

AdaBoost: обобщенная версия.

Имеется T слабых классификаторов h_t, t $\in \{1,...,T\}$.

1. $w_1(i) = \frac{1}{m}, i = 1,...,m.$
2. Для каждого $t = 1,...,T$:
   a. Находим классификатор h_t, минимизирующий взвешенную ошибку $w_t(i)$:
   b. $h_t = \arg\min_{h_j \in H} \varepsilon_j$, где $\varepsilon_j = \sum_{i=1}^{m} w_t(i)[h_j(x_i) \neq y_i]$, ;
      если $\varepsilon_j \geq 0,5$ выход .
   c. Назначаем для h_t вес $\alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$, где ε_t - значение ошибки из шага 2b .
   d. Обновляем веса для данных: $w_{t+1}(i) = \frac{[w_t(i)\exp(-\alpha_t y_i h_t(x_i))]}{Z_t}$, где Z_t такое, что $w_{t+1}(i)$ является распределением, то есть $\sum_{i=1}^{m} w_{t+1}(i) = 1$.
3. Получаем итоговый классификатор: $H(x) = sign(\sum_{t=1}^{T} a_t h_t(x))$

Алгоритм AdaBoost применим для построения детектора лиц или его элементов. В этом случае обучающая выборка состоит из изображений содержащих лица и не содержащих. Как правило, область бровей обычно темнее, чем область глаз, а области глаз темнее, чем область переносицы. Исходя из этих соображений, мы можем использовать следующие примитивы, изображенные на рис. 2, в качестве простейших классификаторов в AdaBoost.

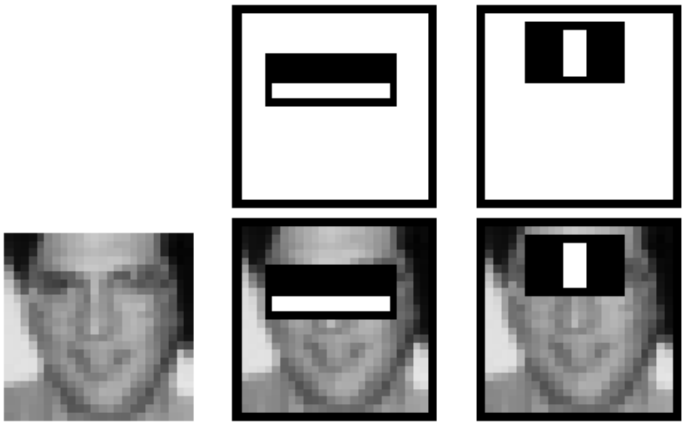


Рис.1 Примитивы, наложенные на изображение лица.

Эти примитивы носят название *примитивы Хаара* в связи с тем, что очень похожи на одноименные вейвлеты. Количество основных типов примитивов – четыре. Два состоящих из двух прямоугольников, один состоящий из трех и один состоящий из четырех. Значение примитива – сумма пикселей в белом прямоугольнике, вычтенная из суммы пикселей в черном прямоугольнике.

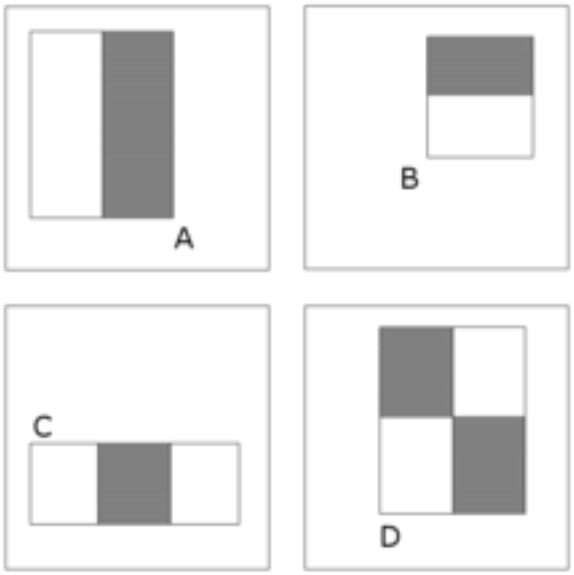
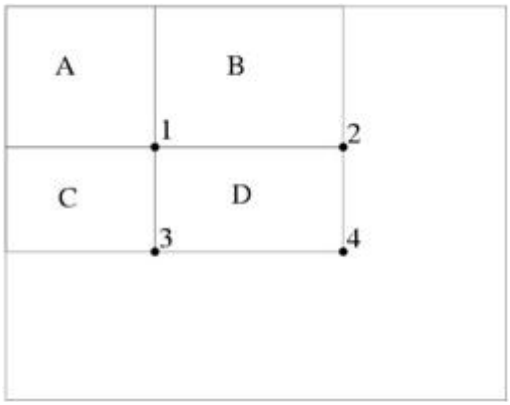


Рис.1 Примитивы, накладываемые на изображения. Значение примитива – сумма пикселей внутри темного прямоугольника минус сумма пикселей внутри белого прямоугольника.

Это очень удобная для распознавания характеристика, но подсчет значений для большого количества примитивов может занять долгое время. Для ускорения

этой операции используется альтернативный способ хранения данных об изображении – интегральное изображение(integral image/summed area table), который позволяет считать примитив любого размера за постоянное время. Суть заключается в том, что изображение хранится не как матрица значений пикселей в каждой точке, а как матрица такого же размера с элементами являющимися суммой всех пикселей левее и выше пикселя, соответствующего данной позиции.



$ii(x, y) = ii(x-1, y) + s(x, y)$
$s(x, y) = s(x, y-1) + i(x, y)$
$ii(x, y)$ – интегральное изображение
$s(x, y)$ – сумма пикселей в ряду левее x
$i(x, y)$ – изображение
$ii(-1, y) = 0 \qquad s(x, -1) = 0$

Рис.3 Интегральное изображение.

Исходя из этих формул, легко найти сумму пикселей в прямоугольнике D = 4 – 3 – 2 + 1.

Количество примитивов, построенное для одного изображения достаточно велико. Например, для изображения размером 24x24 пикселя количество таких примитивов порядка 160 000. Поэтому возникает задача: Как выбрать наиболее подходящие? Для выбора наилучших примитивов можно воспользоваться алгоритмом AdaBoost. Это будет первым этапом в построении главного классификатора.

На этом этапе мы выбираем примитивы, классификаторы которых получают ненулевой вес после работы AdaBoost. Определим *классификатор примитива f*, как функцию:

$$h(x, f, p, \theta) = \begin{cases} 1, \text{если } pf(x) < p\theta \\ 0, \text{иначе} \end{cases}$$

Где x – классифицируемый объект, $f(x)$ – значение примитива данного объекта x, θ – порог классификации, p – полярность классификатора. На основе данного определения на каждом шаге выбирается тот примитив, классификатор которого имеет наименьшее значение ошибки.

AdaBoost: алгоритм для выбора классификаторов примитивов

1. Даны изображения $(x_1, y_1), ..., (x_n, y_n)$, где $y_1 = 0,1$ для отрицательных и положительных образцов соответственно.
2. Назначаем веса $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ для $y_i = 0,1$ соответственно, где m и l число отрицательных и положительных образцов соответственно.
3. Для каждого $t = 1, ..., T$:
   a. Нормируем веса $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$, так чтобы $\sum_{i=1}^{n} w_{t,i} = 1$.
   b. Выбираем классификатор с наименьшей ошибкой $\varepsilon_t = \min_{f,p,\theta} \sum_i w_i \, | h(x_i, f, p, \theta) - y_i |$
   c. Определим классификатор $h_t(x)$: $h_t(x) = h(x, f_t, p_t, \theta_t)$, где θ_t, f_t и p_t значения на которых достигается ε_t.
   d. Обновляем веса: $w_{t+1,i} = w_{t,i} \beta_t^{1-\varepsilon_t}$, где $\varepsilon_t = 0$ если x_i определен корректно, $\varepsilon_t = 1$ иначе, где $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.
4. Получаем итоговый классификатор: $h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{иначе} \end{cases}$ где $\alpha_t = \log \frac{1}{\beta_t}$.

На втором этапе множество примитивов, полученное ранее, используется для формирования каскада классификаторов.

На каждом этапе из множества примитивов выбираются те, которые необходимы для построения классификатора с точностью определения заданной на этапе. Классификатор определяет возможную принадлежность изображения к классу лиц. На каждом последующем этапе формируется более точный и сложный классификатор.

На начальных стадиях обработки изображение проходит через самые простые классификаторы, которые отбрасывают те части изображения, которые явно не являются лицами. Чем дальше изображение проходит, тем более сложные классификаторы используются. При положительном прохождении через весь каскад классификаторов изображение определяется как лицо.

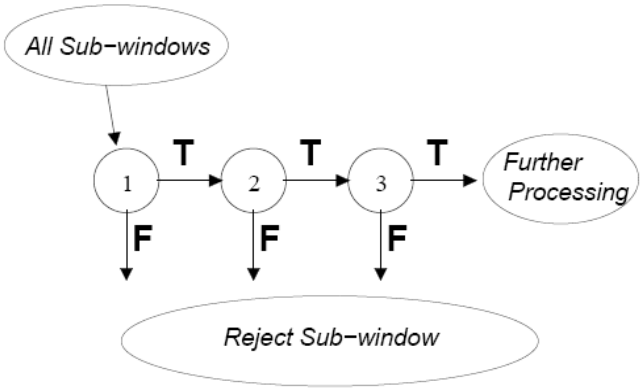
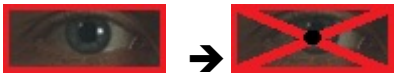


Рис.3 Схема работы каскада классификаторов.

Обучение

Для создания каскада классификаторов были использованы изображения лиц взятые из базы ColorFERET[3] и модули для создания обучающих примеров и тренировки каскадов из библиотеки OpenCV[4]. С помощью утилиты createsamples из 1100 изображений глаз с помощью искажений было получено 5000 изображений глаз. Также для обучения использовались 3020 изображений, не содержащих интересуемых объектов, в нашем случае – глаз. Обучение производилось с помощью утилиты haartraining и состояло из 10 этапов. В параметрах обучения была указана минимальная вероятность правильного определения объекта(min hit rate) равную 0,999, а максимальная вероятность ложного срабатывания(max false alarm) равную 0,5 на каждом этапе. Таким образом, на выходе мы получим каскад, состоящий из 10 классификаторов с точностью $0,999^{10} \approx 0,99$ и вероятностью ложного определения $0,5^{10} \approx 0,00098$ на обучающей выборке.

Для правильного позиционирования трехмерной модели нам необходимо знать положение зрачков человека смотрящего на монитор. Так как при обучении использовались примеры прямоугольной формы, то координаты зрачков можно аппроксимировать центром прямоугольника, выделяющего найденный на рисунке объект.



Тестирование

Для тестирования созданного каскада был использован модуль performance из библиотеки OpenCV. Тестирующая выборка была создана из 50 изображений содержащих интересуемый объект. Ниже приведены результаты тестирования.

```
+================================+======+======+======+
|            File Name           | Hits |Missed| False|
+================================+======+======+======+
|tests/00756_941201_rc.ppm_0000_0|     1|     0|     0|
+--------------------------------+------+------+------+
|tests/00756_941201_rc.ppm_0000_0|     0|     1|     2|
+--------------------------------+------+------+------+
|tests/00767_941201_rb.ppm_0000_0|     1|     0|     3|
+--------------------------------+------+------+------+
|tests/00767_941201_qr.ppm_0000_0|     1|     0|     2|
+--------------------------------+------+------+------+
...........
+--------------------------------+------+------+------+
|                          Total|    43|     7|    79|
+================================+======+======+======+
```

```
Number of stages: 10
```

```
Number of weak classifiers: 691
```

```
Total time: 7.862000
```

Для начальной конфигурации каскада довольно неплохой результат. В дальнейшей разработке планируется увеличить число этапов до 30, а также расширить обучающую выборку для создания более точного каскада классификаторов.

Заключение

- изучен алгоритм AdaBoost и техники бустинга
- изучена библиотеки OpenCV
- собраны и приготовлены данные для тренировки, полученные из [3]
- построен детектора глаз для дальнейшего использования при разработке псевдотрехмерной графики

Ссылки

[1] Yoav Freund, Robert E. Schapire. "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.
[2] Paul Viola, Michael Jones. “Robust Real-Time Face Detection”, 2003.
[3] The Color FERET Database. http://face.nist.gov/colorferet/
[4] OpenCV library. http://opencv.willowgarage.com/wiki/