Министерство образования и науки РФ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования «НИТУ МИСиС»
Институт ИТАСУ
Кафедра Инженерной кибернетики

Курсовая работа
по курсу «Методы Обработки Изображений»
на тему: «Легковесный детектор элементов рекламы»

Выполнил
Студент группы
БПМ-16-2
Фадеев А.Ю.

Проверил:
доц., к.т.н. Полевой Д.В.

Москва 2020

Оглавление

1. Введение

В данный момент на большом количестве веб-страницах основным способом монетизации является рекламный контент. Множество сайтов злоупотребляют использованием вставок, размещая их в большом количестве и/или размещая отвлекающие вставки. Из вышесказанного следует, что задача обнаружения и скрытия рекламного контента является актуальной. Цель этой работы состоит в разработке программного модуля для поиска приблизительного положения таких вставок на изображении.

2. Основная часть

2.1 Введенные обозначения

Введем несколько обозначений:

- Обозначим текущий выбранный элемент рекламы за E
- Обозначим текущий выбранное изображение за S
- Обозначим множество особых точек на E, которые с чем-либо "сматчены"за pts_E, а на S – за pts_S.
- Обозначим центр масс множества pts_E за $cent_E$, а центр масс pts_S – за $cent_S$.
- Общее количество элементов рекламы обозначим за cnt_{ads}
- Количество обнаруженных элементов рекламы обозначим за $cnt_{discovered}$
- Количество корректных матчингов, которые нашли не элемент рекламы, обозначим за cnt_{wrong}
- Количество корректных матчингов обозначим за cnt_{match}

2.2 Формат входных и выходных данных

В директории ads:

Поддиректория	Что хранит
pages	Исходные изображения.
md_pages	Изображения с размеченными на них документами.
elements	Элементы рекламы – кусочки изображений, по которым можно определить нахождение рекламы на изображении.
markdown	находятся текстовые файлы, в которых содержится информация о разметке.
result	результат – изображения с обозначенными на них распознанными элементами рекламы, а также в statistics.txt

Формат данных в файлах поддиректории markdown:

- В первой строке число n - количество прямоугольников.
- В следующих n строках – информация о прямоугольниках.
- В очередной строке находятся 4 числа: координаты x, y прямоугольника, ширина и высота прямоугольника.

Для того, чтобы изменить место, откуда будут считываться данные, необходимо поменять в файлах `main_app.cpp` и `main_md.cpp` переменную `dir_path`.

2.3 Методика оценки качества решения

Критериями качества выбраны 2 характеристики:

- Доля обнаруженных элементов рекламы $= \frac{cnt_{discovered}}{cnt_{ads}}$.
- Доля неверных "матчингов"$= \frac{cnt_{wrong}}{cnt_{match}}$.

2.4 Сборка

Проект собирается через CMake. Требуется иметь библиотеку OpenCV. Возможно, придется "докидывать"файлы `.dll` библиотеки в директорию с проектом, либо прописывать путь до них в **PATH**.

2.5 Разметка

Исходные изображения размечаются вручную.
Например, на изображении

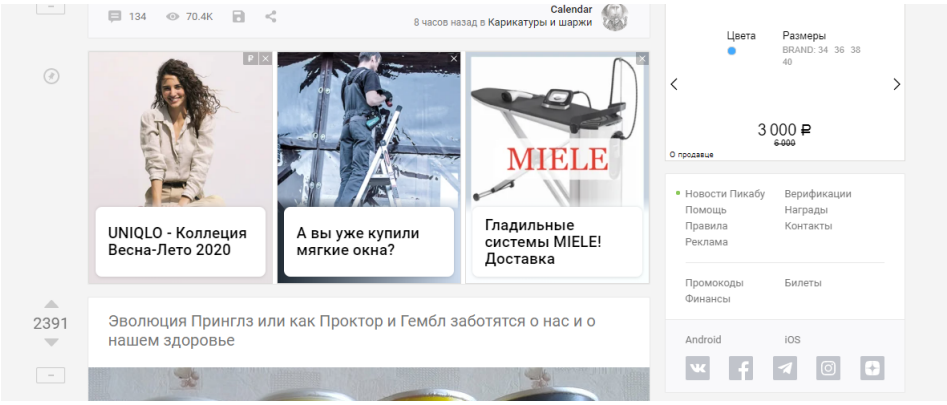

Рис. 2.1. Исходное изображение

Можно выделить такие элементы рекламы:

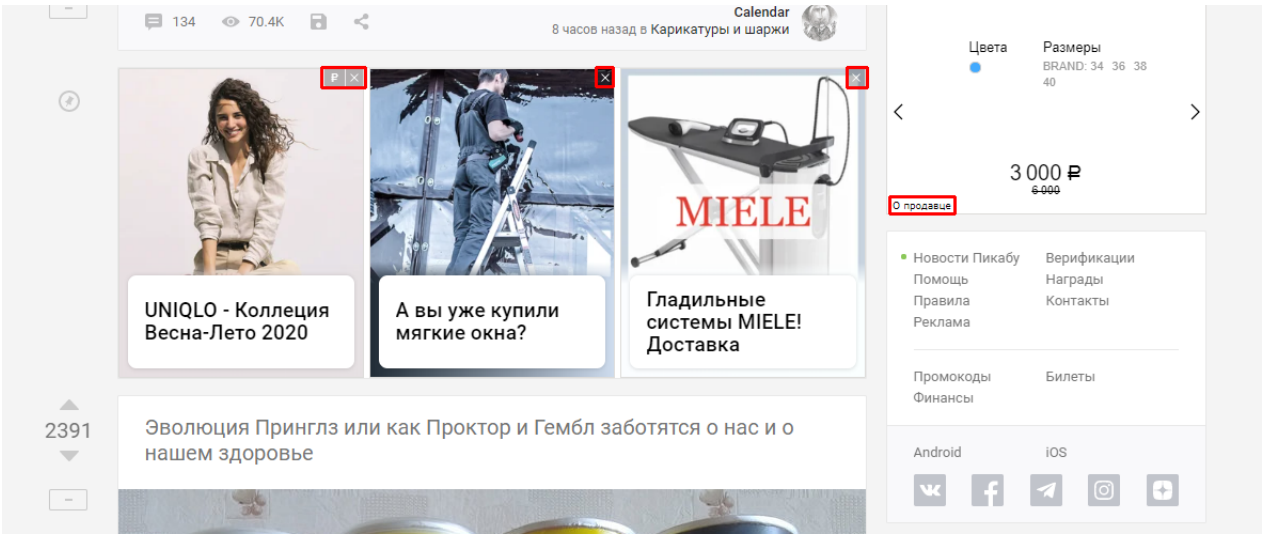

Рис. 2.2. Размеченное изображение

Информация о этих элементах сохраняется, в текстовом виде (информация о прямоугольниках на изображении):

```
[20 x 20 from (791, 58)]
[17 x 19 from (556, 58)]
[42 x 20 from (299, 58)]
[62 x 17 from (831, 179)]
```

Рис. 2.3. Прямоугольники

А также в виде фрагментов изображения:

Рис. 2.4. Элементы

Для этого используется программа из Приложения А.

2.6 Описание алгоритма

Исходные изображения и все элементы рекламы проходят через SIFT, который ищет в них особые точки.

Примеры:

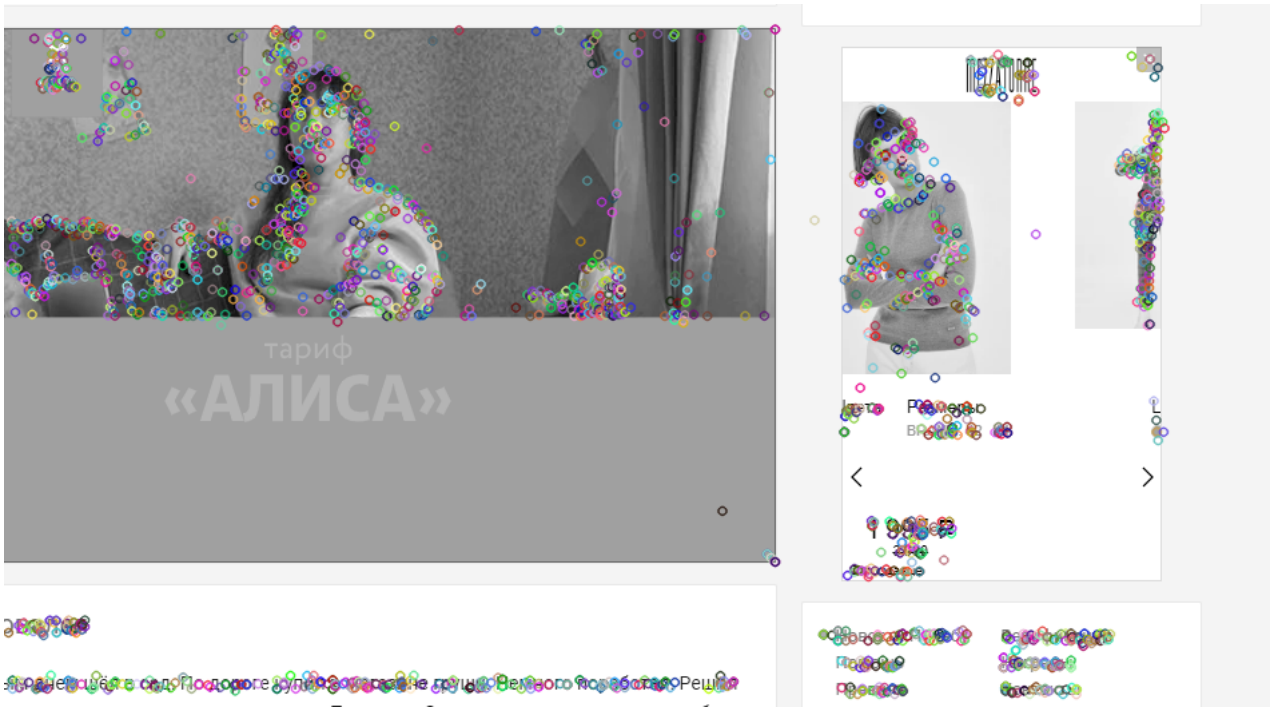

Рис. 2.5. Особые точки на изображении

Рис. 2.6. Особые точки на элементе рекламы

Матчинг производится с помощью алгоритма FLANN.
Далее матчинги проходят проверку на корректность.
Проверка на корректность состоит из нескольких этапов:

1 Хотя бы $P\%$ особых точек E должны быть "смэтчены"с какой-либо особой точной на S (в данной работе $P = 75$).

2 Элемент E, если и присутствует на S, то увеличен не более, чем в K раз (в данной работе $K = 3$).

Математически, это выглядит так:

$$K \times \rho(pts_{Ei}, cent_E) \geq \rho(pts_{Si}, cent_S), \forall i \tag{2.1}$$

3 Углы поменялись не слишком сильно:

$$angle((pts_{Ei} - cent_E), (pts_{Si} - cent_S)) \leq \varepsilon, \forall i \tag{2.2}$$

В данной работе $\varepsilon = \pi/2$.

Если матчинг выполняет все 3 условия, то он считается корректным. При этом, если $cent_page$ попал внутрь какого-либо элемента рекламы, то этот элемент считается обнаруженным. В обратном случае – увеличивается cnt_{wrong}.

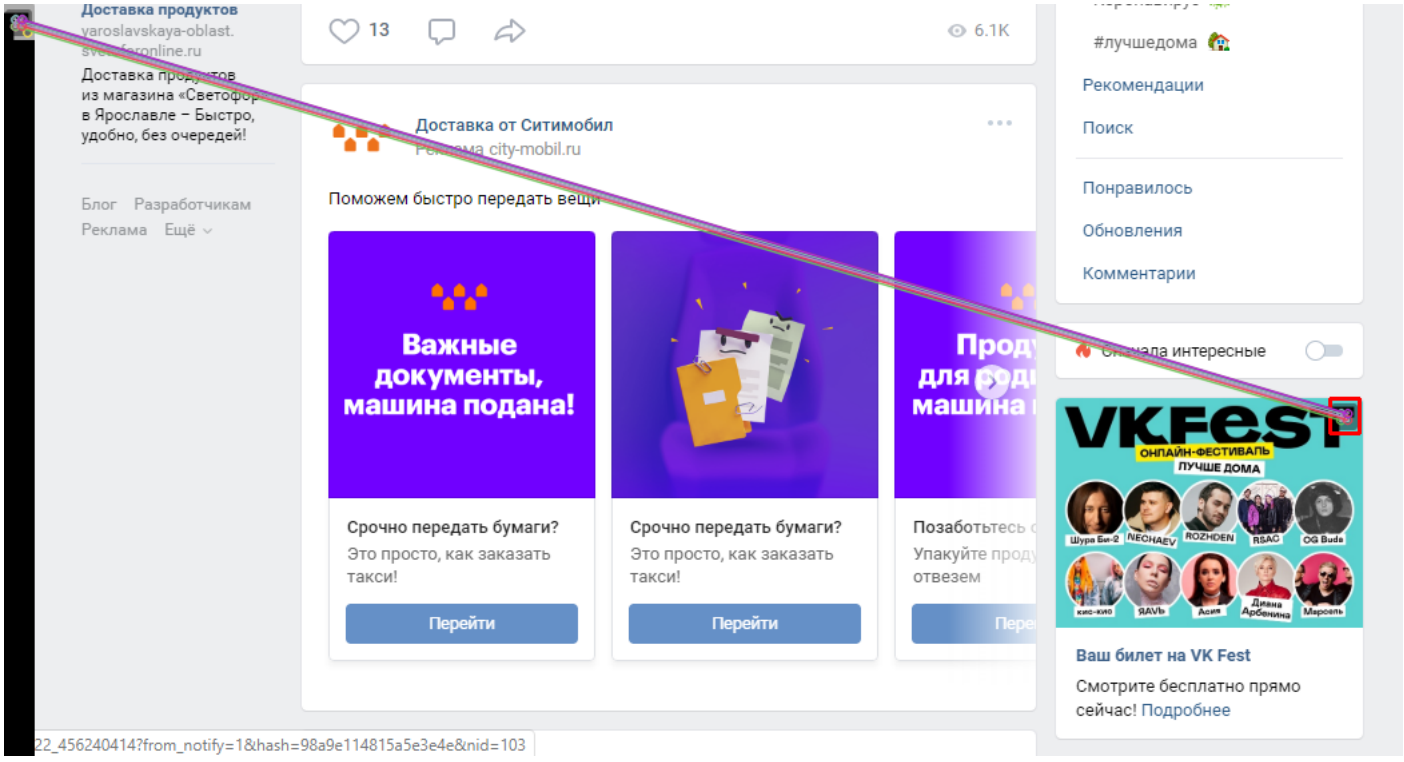

Рис. 2.7. Пример правильного корректного матчинга

2.7 Результаты

Итоговая статистика для сгенерированной выборки, она находится в файле statistics.txt.

- Количество найденных элементов – `12/15`.
- Количество неправильных матчингов (которые привели не в элемент рекламы) – `0/17`.

Пример обработанного изображения, который сохраняется в поддиректорию result. Фиолетовыми кругами отмечены обнаруженные элементы.

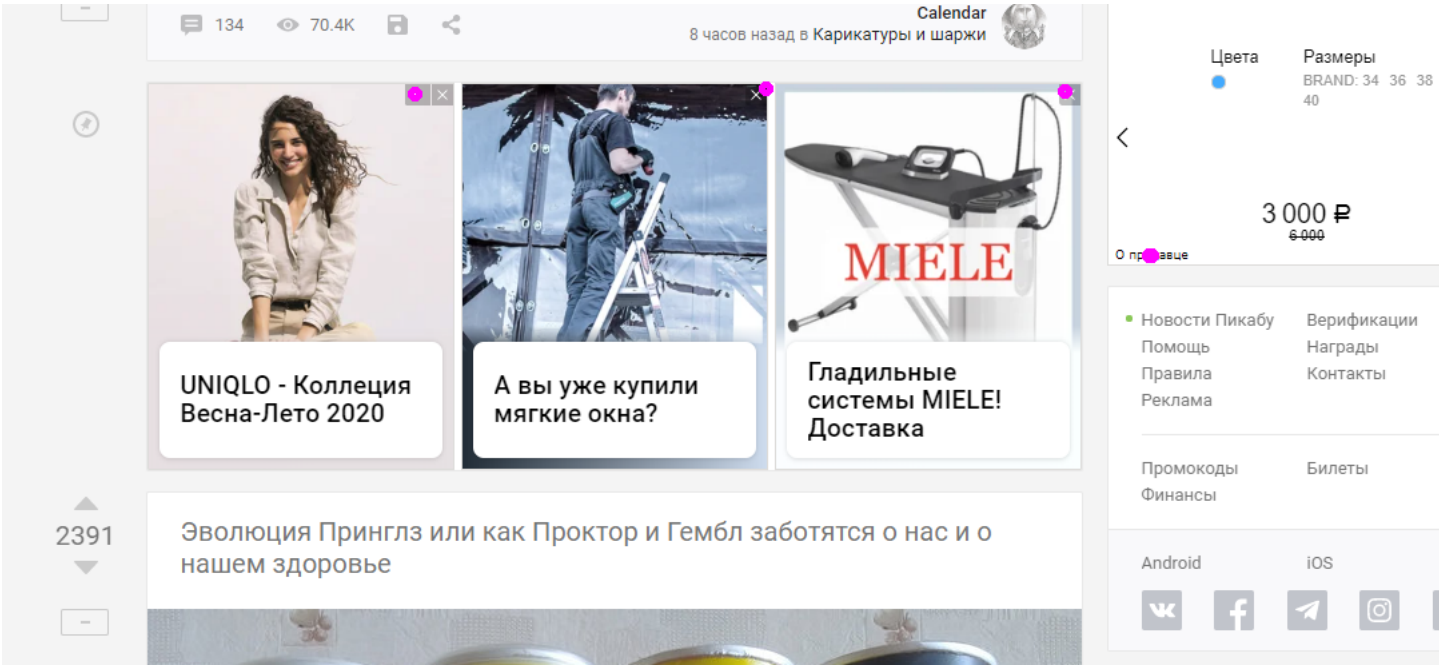

Рис. 2.8. Пример правильного корректного матчинга

3. Приложение

3.1 Приложение А: ссылка на Github.

https://github.com/GhostKicker/imgproc_coursework