

Отчёт по задаче локального автоуточнителя уголков проективно искаженной зоны

Александр Пичугин

6 октября 2025 г.

Цель

Разработка локального автоуточнителя уголков проективно искаженной зоны для улучшения сегментации штрих- и QR-кодов.

Задача

Реализовать автоуточнитель, оценить его качество и проанализировать результаты.

Вход и выход задачи

- **Вход:** RGB-изображение, содержащее предварительно сегментированные штрих-коды/QR-коды
- **Выход:** Координаты уточнённых уголков.

Критерии оценки качества

В качестве метрики оценки качества было взято изменение NCE (Normalized Corner Error).

Формула:

$$NCE = \frac{1}{4} \sum_{i=1}^4 \frac{\|p_i - g_i\|_2}{S},$$

где p_i – предсказанная вершина (уголок), g_i – разметка (ground truth), S – масштаб (квадратный корень из площади четырехугольника, образованного уголками). В свою очередь, изменение NCE показывает, насколько изменяется качество сегментации (а точнее, поиска уголков) при использовании локального автоуточнителя.

Данные

- Собственный датасет аннотированных изображений. Датасет состоит из изображений формата png или jpg, каждое изображение – фотография бар-кода/QR-кода, которые находятся в разных местах фотографии и произвольно ориентированы. Качество изображений разнится от размытого до четкого. Баркоды и QR-коды могут быть проективно искажены.
- Формат аннотаций: json с четырёхугольниками и метками классов. Разметка происходила по описанным правилам.

```
{  
    "objects": [  
        {  
            "index": 0,  
            "data": [  
                [1120, 1965],  
                [1388, 2027],  
                [1299, 2195],  
                [1024, 2178]  
            ],  
            "tags": [],  
            "type": "qr"  
        }  
    ],  
    "size": [3024, 4032]  
}
```

Рис.1. Пример разметки json

Описание метода

TLDR

Для каждой стороны исходного сегментированного прямоугольника берём узкую ленту вокруг этой стороны, извлекаем граничные точки из результата Canny внутри ленты, аппроксимируем сторону прямой методом ортогональной регрессии, затем получаем вершины как пересечения соседних прямых. В конце упорядочиваем вершины и возвращаем результат.

Алгоритм

Вход. Изображение + сегментация объекта (такой же json-формат, как и у разметки (описано выше)).



Рис. 1: Иллюстрация: вход алгоритма

For i = 1..N do:

1. Переводим изображение в один канал;



Рис. 2: Иллюстрация: перевод изображения в один канал

2. Применяем небольшое гауссово сглаживание, чтобы снизить шум;



Рис. 3: Иллюстрация: гауссово сглаживание изображения с окном 3.

3. Формируем по узкой полосе вокруг каждой из сторон прямоугольника;

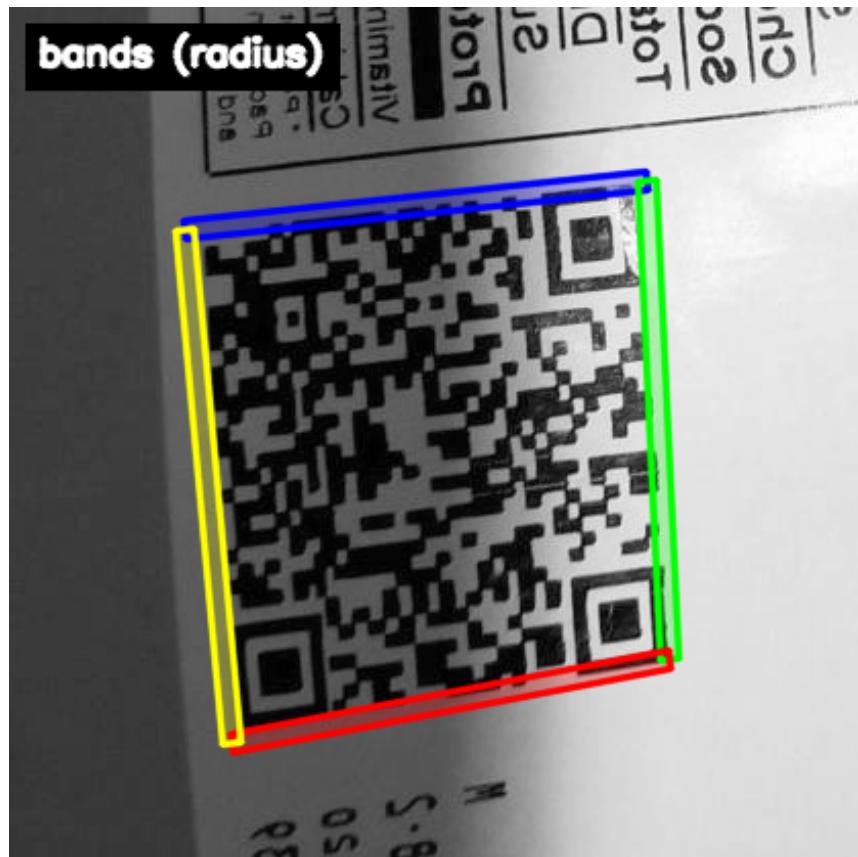


Рис. 4: Иллюстрация: полоски радиуса 5.

4. Выбираем пиксели краев в границах каждой из лент, используя Canny;



Рис. 5: Иллюстрация: canny с параметрами 30, 150.



5. Для каждой стороны ищем центр масс точек, затем оцениваем направление скорректированной стороны через PCA/SVD либо иную интерполяцию;

Примечание. В репозитории была имплементирована аппроксимация `cv2.fitLine(..., distType=cv2.DIST_L2, ...)` – total least squares, что является эквивалентом 1D-PCA.



Рис. 7: Иллюстрация: точки canny, взятые на пересечении с полосой на правой грани, с аппроксимированной прямой (обозначена желтым цветом).

6. Наконец, считаем пересечения получившихся прямых.



Рис. 8: Иллюстрация: пересечение интерполированных прямых.

End for

Результаты

Программное и аппаратное обеспечение

- Датасет QR и баркодов (qrcode, azteccode, pdf417, datamatrix, code128, code39, ean13, ean14, ean8, issn, microqrcode, upca, pzn).
- Собственно подготовленный репозиторий кода.
- Аппаратный ускоритель Apple MPS.

Метрики

Были произведены эксперименты на датасете собственной разметки. Размер датасета – 121 изображение.

Для оценки качества к GT (ручной разметке) был применён гауссовский шум с параметрами $(0; 6 * 416 * 416 / L / W)$, где L и W – длина и ширина каждого изображения в пикселях. Полученные изображения в иллюстрациях называются SEGMENTED.

Изображения, полученные в результате работы Алгоритма, имеют обозначение REFINED.

Для аугментации запущённых данных, было зафиксировано 5 различных случайных сидов. В итоге получено 480 изображений. Результаты представлены в Таблице 1. Примеры работы алгоритма см. на рисунках 9, 10, 11.

Метрика	Среднее	Дисперсия
NCE (before)	0.008	0.004
NCE (after)	0.003	0.001

Таблица 1: Результаты экспериментов



Рис. 9: Пример работы алгоритма на Штрих-коде



Рис. 10: Пример работы алгоритма на QR-коде



Рис. 11: Пример работы алгоритма на QR-коде

Анализ сложных случаев

- Для некоторой части штрихкодов, в том числе, "дизайнерских" не было получено требуемое автоточнение.



Рис. 12: Отсутствие улучшения на дизайнерском штрих-коде с асимметричным скосом

- Алгоритм оказался чувствителен к элементам (резким изменениям интенсивности), близким к границам размечаемой зоны. Например, на рис. 13 алгоритм захватил границу поля для штрих-кода в качестве верхней стороны прямоугольника, а также не сместился к истинной нижней границе прямоугольника ввиду наличия цифр.



Рис. 13: Проблема: чувствительность алгоритма к элементам, близким к границам размечаемой зоны

- Необходимость тонкой настройки гиперпараметров алгоритма. Напри-

мер, слишком большая ширина полоски не позволяла получить высокую точность, так как было получено много ложных краевых точек. В свою очередь, маленькая ширина полоски в некоторых случаях не позволяла получить достаточное количество краевых точек для точной аппроксимации.

Дальнейшее развитие

- Выбор нескольких репрезентативных метрик для автоматического контроля качества;
- Тестирование работы алгоритма на много большем размере датасета ($\gg 100$ изображений) для увеличения статистической значимости экспериментов.
- Подбор гиперпараметров, таких, как ширина полоски, пороги Canny, количество итераций алгоритма, при помощи GridSearch, в идеале, эти гиперпараметры должны зависеть от номера итерации. Также следует подобрать порог раннего останова алгоритма.
- Внедрение детектора цифр с целью исключения их из точек границ, получаемых Canny, так как в некоторых случаях это мешает корректной работе алгоритма.
- Увеличение стабильности работы алгоритма на некоторых дизайнерских штрих-кодах. Например, локально (вдоль границ) аппроксимировать линии штрих-кода и использовать это для "доворота" линии. Ожидается, что это должно помочь с проблемами наподобие 12.