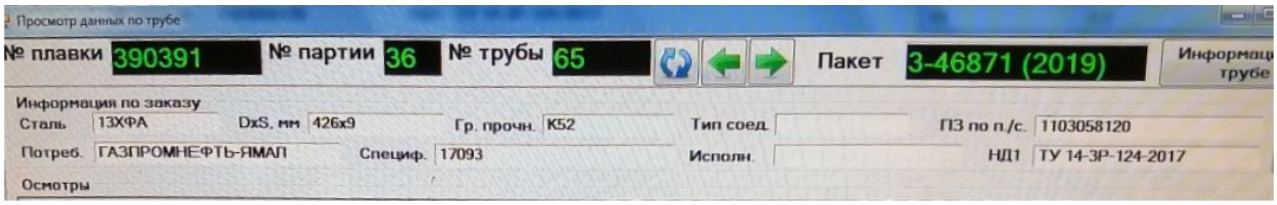
Задание

Дана фотография плохого качества.



1. Вытащить, по возможности, всю информацию из полей.
2. Если что-то не получится вытащить, описать причину неудачи.

Извлечение информации

Для начала допущу, что местоположение и размеры полей на изображении будут постоянными. Этого можно добиться простым printscreen-ом или, для особых случаев, захватом видеопотока с рабочего стола.

Выделю все поля на изображении и получу двоичную маску:

Далее разделю её на составные части (13 частей) и сохраню в отдельные файлы. Для отсутствия потери качества буду использовать расширение png для всех файлов изображений.

Пример одной маски с именем mask1.png

Далее заполню файл description.txt

---
mask1.png
№ плавки
---
mask2.png
№ партии
---
mask3.png
№ трубы
---
mask4.png
Пакет
---
mask5.png
Сталь
---
mask6.png
DxS, мм
---
mask7.png
Гр. прочн.
---
mask8.png
Тип соед.
---
mask9.png
ПЗ по п./с.
---
mask10.png
Потреб.
---
mask11.png
Специф.
---
mask12.png
Исполн.
---
mask13.png
НД1

Здесь создана ассоциация между именем файла маски и именем соответствующего поля. По большей части эта ассоциация влияет только на то, как программа будет называть распознанные данные, поэтому эти имена полей можно менять.

Процесс обработки изображения

Предварительно вычисляю параметры масок и вырезаю нужное изображение из исходного:

```
(x,y,w,h) = dataset[name]['xywh']
image = data[y:(y+h), x:(x+w)]
```

Произвожу увеличение изображения и бикубическую интерполяцию, что позволяет сгладить границы увеличенного изображения и получить более высокое качество распознавания текста при данных низкого разрешения:

```
image = cv2.resize(image, None, fx=resize, fy=resize, interpolation=cv2.INTER_CUBIC)
```

Убираю шумы серого и белых цветов, вызванные матрицей монитора. Выделяю значимые изменения цвета по динамическому порогу Оцу, чтобы выделить текст. Инвертирую светлый текст на черном фоне:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray = cv2.threshold(gray, gray_border, 255,
                     cv2.THRESH_TRUNC)[1]
gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
if (dataset[name]["order"] < 4):
   gray = cv2.bitwise_not(gray)
```

Настраиваю модуль Tesseract и вызываю его

Общее описание команд

«-l rus» - устанавливаю русский язык

«--oem 2» - Установка «ocrenginemode» (ссылка с подробным описанием)

2 - Запустите оба (Tesseract и Cube) и объедините результаты - лучшая точность

«--psm 7» - Установка «pagesegmode» (ссылка с подробным описанием)

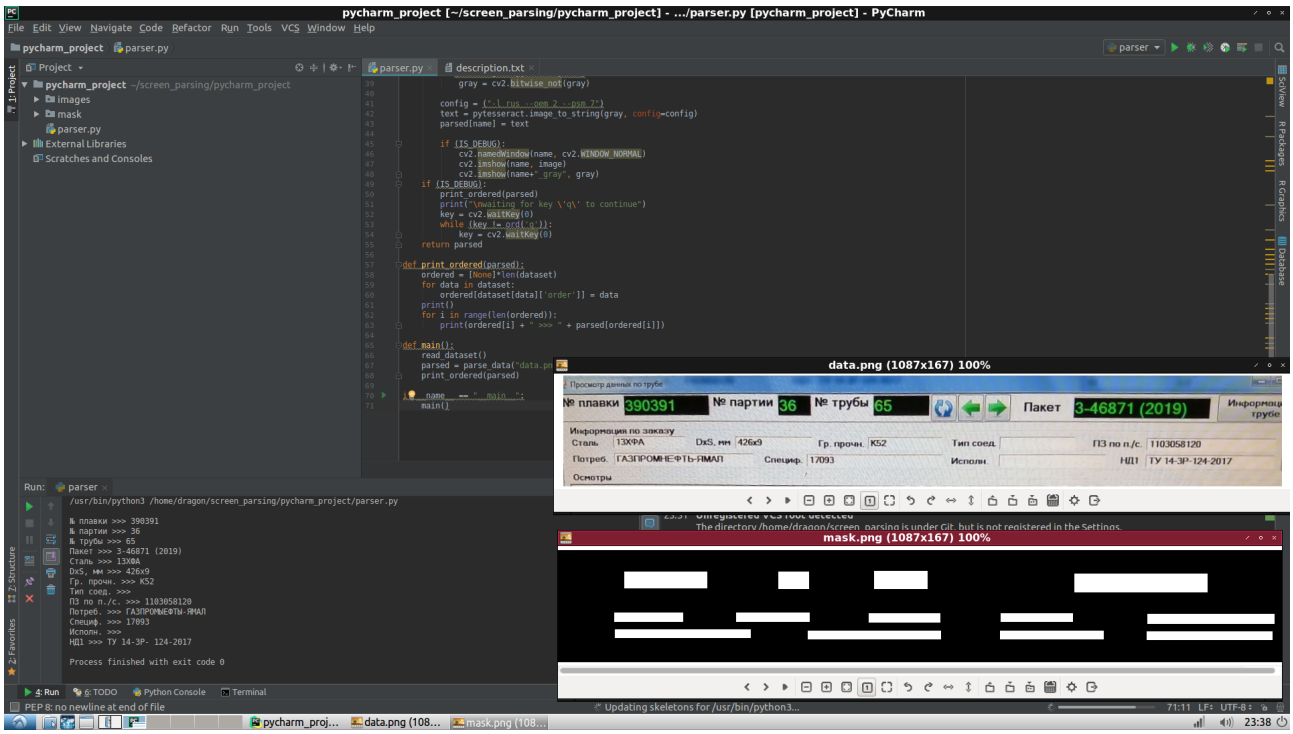
7 - Обработать изображение как одну текстовую строку.

```
config = ("-l rus --oem 2 --psm 7")

text = pytesseract.image_to_string(gray, config=config)
```

Оценка точности

Пример работы программы:



№ плавки >>> 390391

№ партии >>> 36

№ трубы >>> 65

Пакет >>> 3-46871 (2019)

Сталь >>> 13ХФА

DxS, мм >>> 426x9

Гр. прочн. >>> K52

Тип соед. >>>

ПЗ по п./с. >>> 1103058120

Потреб. >>> ГАЗПРОМЫЕФТЫ-ЯМАЛ

Специф. >>> 17093

Исполн. >>>

НД1 >>> ТУ 14-3Р- 124-2017

Несерьёзная ошибка - появление лишних пробелов.

Пример «ТУ 14-3Р- 124-2017» вместо «ТУ 14-3Р-124-2017»

Серьёзная ошибка — ошибка распознавание текста.

Пример «ГАЗПРОМЫЕФТЫ-ЯМАЛ» вместо «ГАЗПРОМНЕФТЬ-ЯМАЛ»

Но давайте же посмотрим, как видит наши данные сама программа...

При таком исходном разрешении изображения не удивительно, что символы распознаются неверно.

А вот это уже результат работы моих фильтров. Ситуация улучшилась, но разобрать и отделить некоторые символы по-прежнему сложно — не хватает качества изображения.

Выводы

Из исходных данных удалось вытянуть достаточно информации для успешного распознавания чисел и простого текста. Однако для успешного распознавания любых текстов необходимо значительно более высокое качество изображения. Рекомендуется отправлять программе printscreen окна в формате png.

Исходный код и инструкции по установке:

github.com/d3dx13/screen_parsing