

# **Финальный отчёт**

## **Постановка задачи**

Написать утилиту для программной генерации изображений бар кодов приближенных к реалистичным.

## **Ожидаемый результат**

Программа которая генерирует изображение с бар кодами и разметку расположения бар кодов как в VIA (VGG Image Annotator)

## **Шаги решения задачи**

1. Выбрать библиотеку для генерации бар кодов (были протестированы 3 библиотеки и был выбран `treepoem` как с самой широкой поддержкой бар кодов)
2. Аффинная деформация - реализована случайная аффинная деформация бар кода, в эмпирически подобраны параметры нормировки чтобы изображение не съезжало с экрана.
3. Написать код генерации разметки в формате VGG
4. Добавление произвольного заднего фона и произвольного количества бар кодов: для этого я генерирую маску и деформирую её вместе с бар кодом чтобы после этого корректно наложить бар код на изображение
5. Случайная деформация перспективой. Возникла сложность с параметрами нормировки чтобы изображение не слишком деформировалось, было использовано решение с эмпирическими коэффициентами для суммы случайных матриц
6. Модификация случайной деформации перспективой для гарантии читаемости бар кода - деформация перспективой генерируется на основе перемещения углов бар кода, что позволяет гарантировать его читаемость
7. Добавлены аугментации из `augraphy`
8. Автоматизирована генерация входных данных для самых популярных бар кодов для генерации бар кодов в `treepoem`

## **Результат**

Программа которая генерирует изображение с случайно деформированными перспективой бар кодами с различными аугментациями бумаги плохого качества и разметку расположения бар кодов как в VIA (VGG Image Annotator) на основе конфига и входного изображения

## Запуск

### Инструкция

Чтобы сгенерировать изображение с бар кодами нужно запустить скрипт generator.py и передать в него параметр -c= файл с конфигом:

```
python .\generator.py -c="test_conf.json"
```

## Структура конфига

Конфиг - это json в файл в котором нужно указать следующие поля:

- `barcode_types` - список из типов бар кодов (список поддерживающих генерацию контента можно найти в [таблице](#Таблица бар кодов для которых поддерживается автоматическая генерация контента)] `data_generator.py`, а список всех доступных можно найти в документации `treepoem`)
- `name` - имя файла в который сохранится картинка и в который сохранится разметка  
Опционально можно добавить:
  - `barcode_contents` - список строк или чисел которые будут записаны в бар кодах
  - `source_img` - путь до изображения поверх которого будет рисоваться бар коды
  - `augmentations` - список аугментаций которые будут применены к бар кодам, список аугментаций можно найти в [таблице](#Таблица доступных аугментаций)
  - `source_scale` - число > 0, масштабирует изображение, уменьшая или увеличивая вероятность пересечения бар кодов
  - `noise_scale` - число > 0, степенью проективных деформаций баркодов, см. примеры генерации
  - `barcode_scales` - список чисел > 0 для каждого из бар кодов, задает во сколько раз будет увеличен или уменьшен каждый из них на итоговом изображении

## Приложение

### Пример запуска

Запуск `python .\generator.py -c="test_conf.json"` с конфигом

```
{  
    "name": "test",  
    "barcode_types": ["ean13", "azteccode", "aztecrune", "qrcode", "code93",  
    "microqrcode", "datamatrix"],  
    "augmentations": ["Folding", "BadPhotoCopy", "LightingGradient"],  
    "source_img": "./example.jpg",  
    "source_scale": 0.4  
}
```

генерирует следующее изображение:



После импорта в VIA выглядит вот так:

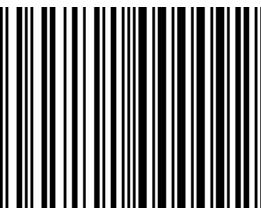


Вариации параметра `source_scale` выглядят следующим образом



## Таблица бар кодов для которых поддерживается автоматическая генерация контента

Название	Размерность	Пример контента	Пример генерации
ean13	1d	684258720442	

Название	Размерность	Пример контента	Пример генерации
upca	1d	0451019	
plessey	1d	D1A7CE96933FDC09AC30	
code39	1d	X251S2ZQTUJ	
code93	1d	PPACZE790I8R G\$Z08W4UKRFNSJ/I3\$3%EWHZ	
datamatrix	2d	}GUQ!~%:WhB}4!!NW=kVm!C}qjuAo	
qrcode	2d	EcWT{<Nc-(C[z'Ig%DOT]4{ L#t0	
azteccode	2d	*>3~HKa^a8Ya#,E1)fY_yrWJN=wH'\\&Cv3F	
aztecrune	2d	30	
microqrcode	2d	/NRt	

## Таблица доступных аугментаций

Название	Пример аугментации
----------	--------------------

Название	Пример аугментации
BadPhotoCopy	
BrightnessTexturize	
ColorPaper	
Folding	

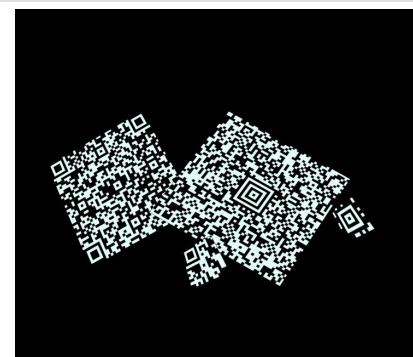
Название	Пример аугментации
LightingGradient	
NoisyLines	
ShadowCast	

## Таблица примеров генераций с разными конфигами

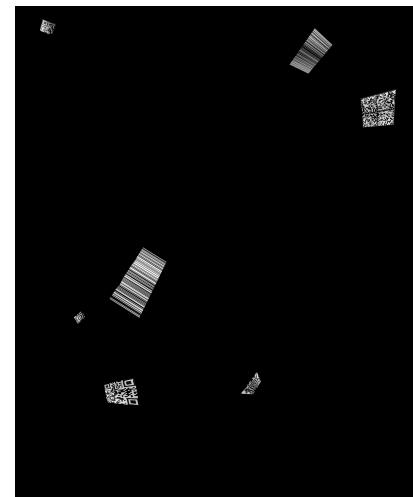
конфиг	результат генерации

**конфиг****результат генерации**

```
{  
    "name": "no_background_2d",  
    "barcode_types": ["datamatrix", "qrcode",  
"azteccode", "aztecrune", "microqrcode"],  
    "augmentations": ["ColorPaper"],  
    "source_scale": 1  
}
```



```
{  
    "name": "no_background",  
    "barcode_types": ["ean13",  
"azteccode", "aztecrune", "qrcode",  
"code93", "microqrcode", "datamatrix"],  
    "augmentations": ["Folding",  
"BadPhotoCopy", "LightingGradient"],  
    "source_scale": 1  
}
```



```
{  
    "name": "bottles_1d",  
    "barcode_types": ["ean13", "plessey",  
"code39", "upca", "code93"],  
    "augmentations": [],  
    "source_img": "./bottles.jpg",  
    "source_scale": 4  
}
```



## конфиг

```
{  
    "name": "bottles",  
    "barcode_types": ["ean13", "azteccode",  
"aztecrune", "qrcode", "code93",  
"microqrcode", "datamatrix"],  
    "augmentations": ["Folding",  
"BadPhotoCopy", "LightingGradient"],  
    "source_img": "./bottles.jpg",  
    "source_scale": 2  
}
```

## результат генерации



```
{  
    "name": "waterfall_2d",  
    "barcode_types": ["datamatrix", "qrcode",  
"azteccode", "aztecrune", "microqrcode"],  
    "augmentations": ["ShadowCast",  
"BadPhotoCopy", "NoisyLines"],  
    "source_img": "./example.jpg",  
    "source_scale": 0.4  
}
```



```
{  
    "name": "different_bar_code_scales",  
    "barcode_types": ["ean13", "ean13",  
"ean13", "ean13", "ean13", "ean13",  
"ean13"],  
    "barcode_scales": [9, 3, 3, 1.5, 1.5,  
1, 0.5],  
    "augmentations": ["ShadowCast",  
"BrightnessTexturize"],  
    "source_img": "./example.jpg",  
    "noise_scale": 0,  
    "source_scale": 0.1  
}
```



**конфиг**

```
{  
    "name": "noise_scale_small",  
    "barcode_types": ["ean13", "ean13",  
"ean13", "ean13", "ean13", "ean13",  
"ean13"],  
    "barcode_scales": [9, 3, 3, 1.5, 1.5,  
1, 0.5],  
    "augmentations": ["ShadowCast",  
"BrightnessTexturize"],  
    "source_img": "./example.jpg",  
    "noise_scale": 0.1,  
    "source_scale": 1  
}
```

**результат генерации**

```
{  
    "name": "noise_scale_mid",  
    "barcode_types": ["ean13", "ean13",  
"ean13", "ean13", "ean13", "ean13",  
"ean13"],  
    "barcode_scales": [9, 3, 3, 1.5, 1.5,  
1, 0.5],  
    "augmentations": ["ShadowCast",  
"BrightnessTexturize"],  
    "source_img": "./example.jpg",  
    "noise_scale": 0.3,  
    "source_scale": 1  
}
```



**конфиг**

```
{  
    "name": "noise_scale_high",  
    "barcode_types": ["ean13", "ean13",  
"ean13", "ean13", "ean13", "ean13",  
"ean13"],  
    "barcode_scales": [9, 3, 3, 1.5, 1.5,  
1, 0.5],  
    "augmentations": ["ShadowCast",  
"BrightnessTexturize"],  
    "source_img": "./example.jpg",  
    "noise_scale": 1,  
    "source_scale": 2  
}
```

**результат генерации**