

# Решение sudoku при помощи компьютерного зрения

Бызова Анна, Дымова Диана

# Описание

Проект реализует автоматическое распознавание чисел на игровом поле sudoku с помощью компьютерного зрения, а затем применяет алгоритм решения sudoku для поиска ответа. Это позволяет не только решать стандартные задачи, но и изучать теорию чисел и алгоритмы.

Задача									Решение								
8	4	5	1	2		7	3	9	8	4	5	1	2	6	7	3	9
7		9	5		4		6	2	7	1	9	5	3	4	8	6	2
3		2	9	8	7			5	4	3	6	2	9	8	7	1	5
	3	6			5	2	8		9	3	6	4	7	5	2	8	1
4	8	1	6	9	2		7	3	4	8	1	6	9	2	5	7	3
	5	7	3	1	8	4	9	6	2	5	7	3	1	8	4	9	6
1	7		2	6		3	4	5	1	7	8	2	6	9	3	4	5
5	9	3	7	4	1	6		8	5	9	3	7	4	1	6	2	8
6		4	8	5		9	1		6	2	4	8	5	3	9	1	7

Задача									Решение								
4		8		6	5			1	4	3	8	9	6	5	2	7	1
5	6	9			2	4	8	3	5	6	9	7	1	2	4	8	3
7		1			8	6		5	7	2	1	3	4	8	6	9	5
	8		2	7	4			6	9	8	3	2	7	4	1	5	6
	4			3	9	8		7	1	4	6	5	3	9	8	2	7
2	7	5			1			3	2	7	5	6	8	1	9	3	4
	9	2		5	3	7		8	6	9	2	1	5	3	7	4	8
	1		8	9	7	5		2	3	1	4	8	9	7	5	6	2
			4	2		3	1	9	8	5	7	4	2	6	3	1	9

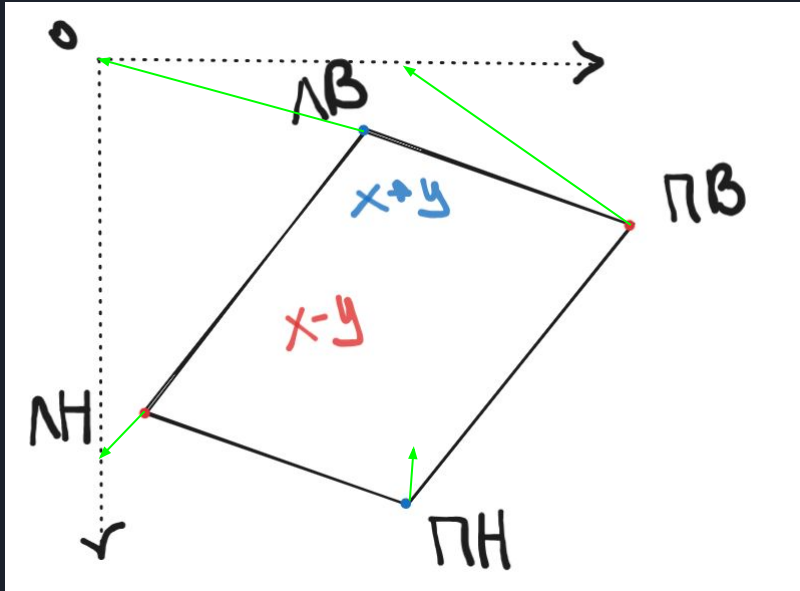
Задача									Решение								
	1			6				5	2	1	7	3	6	4	8	5	9
		6					2		3	5	9	6	7	8	1	2	4
		4		5				1		8	3	4	2	5	9	7	1
					7					1	4	5	9	7	6	3	8
			3		2	8	4		5	7	6	3	1	2	8	4	9
9		2	4							9	8	2	4	3	5	6	7
3	7		8	9			5	6		3	7	1	8	9	2	5	6
		8							7	4	5	8	6	1	3	9	2
		9	5							6	2	9	5	4	7	1	3



Процесс решения мы разбили на несколько этапов:

- Определение границ sudoku и выравнивание поля (OpenCV)
- Распознавание исходных данных (EasyOCR)
- Решение задачи (PuLP)
- Отображение ответа на исходном изображении

# Этап 1. Ориентирование изображения

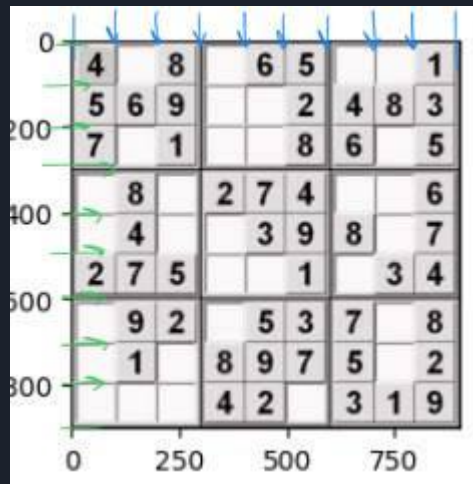


Определим вершины квадрата  
судoku.

верхний левый угол - наименьшая  
сумма координат  
нижний правый - наибольшая сумма  
координат

верхний правый угол - наибольшая  
разность  
нижний левый - наименьшая  
разность

## Этап 2. Подготовка к распознаванию



Разделяем изображение на 9 частей по строкам и на 9 по столбцам, чтобы в дальнейшем распознавать цифры строго внутри ячеек

Для этого используем функцию `np.split`

```
# Разделяем наш sudoku на 9 строк и 9 столбцов и распознаём  
каждое значение  
split = np.split(board, 9, axis=1)  
for col, j in enumerate(split): #  
    digs = np.split(j, 9) # тут разделяем горизонтальные блоки  
на 9 ячеек
```



## Этап 3. Распознавание

Для распознавания чисел используем модель OCR

```
# Загружаем модель OCR (оптическое распознавание символов)
import easyocr
reader = easyocr.Reader(['en']) - создаем функцию чтения изображения
```

С помощью функции распознаем изображение внутри каждой ячейке (циклом) и отображаем его в датафрейм

```
# Распознаём число в ячейке и записываем его в датафрейм и список с координатами
text = reader.readtext(d, allowlist='0123456789', detail=0)
# распознавание цифр из списка 0123456789 без дополнительной информации
if len(text) > 0: # проверяем был ли распознан текст
    df.iloc[row, col] = text[0] # если да - записываем его в датафрейм
    sudoku_map.append([text[0], str(row+1), str(col+1)])
```



## Этап 4. Решение sudoku

Решаем задачу с помощью перебора

По правилам решения sudoku

1/ Каждая цифра (от 1 до 9) в строке должна быть представлена 1 раз

2/ Каждая цифра в столбце должна быть представлена 1 раз

3/ Каждая цифра в квадрате 3 на 3 должна быть представлена один раз

Используем эти ограничения для определения подходящего варианта

```
prob = LpProblem("Судоку", LpMaximize)
```

```
# Создаём Pulp словарь с переменными возможных ответов
```

```
choices = LpVariable.dicts("Choice", (vals, rows, cols), 0, 1,  
LpInteger)
```

## Этап 5. Отрисовываем конечный результат

Пишем решение поверх изображения внутри ячеек  
С помощью функции `imshow()`

Задача

	1			6			5	
		6				2		3
		4		5			1	
				7				
		3		2	8	4		5
9		2	4					
3	7		8	9		5	6	
		8						7
		9	5					

Решение

2	1	7	3	6	4	8	5	9
5	9	6	7	8	1	2	4	3
8	3	4	2	5	9	7	1	6
1	4	5	9	7	6	3	8	2
7	6	3	1	2	8	4	9	5
9	8	2	4	3	5	6	7	1
3	7	1	8	9	2	5	6	4
4	5	8	6	1	3	9	2	7
6	2	9	5	4	7	1	3	8