#Intelligent Placer

##Содержание:

1. [Что делает код в этом репозитории?](#pWhat)

2. [План (Подход к решению задачи)](#pPlan)

3. [Требования к фото](#pPhoto)

4. [Требования к массиву координат](#pCoor)

5. [Сбор данных](#pData)

6. [Усложнения](#pHard)

7. [Словарь обозначений](#pDict)

##Что делает код в этом репозитории? <a id ="pWhat"></a>

По данным на вход фотографиям нескольких премедотов на белом листе бумаги A4 и многоугольнику заданному по правилу [(1)](#pCoor), этот код выдает true если описанные предметы [ПЦВМ](#PCVM), false в противном случае.

Код оформлен в виде python-библиотеки **\_\_intelligent\_placer\_lib\_\_**, которая поставляется каталогом **\_\_intelligent\_placer\_lib\_\_** с файлом **\_\_intelligent\_placer.py\_\_**, содержащим функцию - точку входа

```

def check\_image(<path\_to\_png\_jpg\_image\_on\_local\_computer>[, <poligon\_coordinates>])

```

То есть так, что работает код:

```

from intelligent\_placer\_lib import intelligent\_placer

def test\_intelligent\_placer():

    assert intelligent\_placer.check\_image(“/path/to/my/image.png”)

```

Также содержиться воспроизводимый **\_\_intelligent\_placer.ipynb\_\_**, содержащий репрезентативные примеры работы алгоритма с оценками качества его работы и их визуализацией [(3)](#3).

**### Описание директорий**

\* photos/ - фотографии предметов на белом листе бумаги A4, на темном однородном фоне, удовлетворяющие [требованиям](#pPhoto). В формате 'n.jpg' [(n - номер предмета)](#pData).

    \* photos/Oxy/ - фото для объяснения введенной [СК](#pCoor).

    \* photod/TestResExpect/ - конфигурации предметов для которых писались

    [тесты](#pData). С которыми сравниваем результаты алгорима.

\* tests/ - папка для п. [Сбор данных](#pData), в которой лежат файлы testVer1.xlsx и testVer1.csv содержимое их идентично, две копии нужны для того, чтобы если один формат дает баг можно было использовать другой. (Файлф .csv сохраненные на моем ПК не всегда нормально открываются на другом устройстве).

    \* tests/testVer0/ - папка с тестами в другом формате не особо удобном, но я их оставил возможно еще пригодяться.

\* docs/ - содержит файл Intelligent Placer.docx. Его содержимое идентично README.md, нужен для проверки правил русского языка в этом файле. Почему в отдельной папке? Чтобы не мозолил глаза, иконка папки приятней иконки ворд документа.

**## План (Подход к решению задачи) <a id ="pPlan"></a>**

**#### Часть 1. Начальные данные.**

Программа находит особые точки и границы предмета (детектор Харриса/Нейросеть). Программа находит границы листа бумаги. Вычисляет координаты особых точек предмета в [плоскости листа](#pCoor).

**#### Часть 2. Алгоритм расположения предмета:**

Совмещаем предметы по [ОТ](#OT) с [ОТ](#OT) многоугольника. Также для каждой совмещенной пары проверяем повороты с шагом 5° ([ВИВ](#VIV)).

Для предмета - множество особых точек определено в 'Часть 1'.

Для многоугольника выделяем переборное множество точек

\* Его вершины.

\* Центры соторон.

\* (Центры половин сторон).

Когда мы рассматриваем предмет номер n в алгоритме.

Пусть $W$ - множество [ОТ](#OT) многоугольника,

$V\_k$ -  множество [ОТ](#OT) k-го предмета, который [ПЦВМ](#PCVM) .

Эти множества содержат координаты вершин в плоскости листа бумаги.

Тогда множество особых точек многоугольника:

$$ W = W \cup \left(\cup\_{k=1}^{n - 1} V\_k \right)$$

Как определить, что предмет поместился или нет?

Берем [ОТ](#OT), для которой на данном шаге ищем поворот предмета при котором он [ПЦВМ](#PCVM). Берем следующую после нее [ОТ](#OT) предмета. Мы знаем ее координаты в плоскости листа бумаги и можем определить будет она находиться в многоугольнике или нет.

\* Если она не лежит, переходим к новому углу поворота, либо меняем точку.

\* Если лежит переходим к следующей [ОТ](#OT), пока не совершим полный круг. В этому случае предмет [ПЦВМ](#PCVM).

**#### Часть 3. Завершение.**

Вывод true, false по результатам работы алгоритма.

**## Требования к фото. <a id ="pPhoto"></a>**

1. На фото есть лист бумаги, предмет и фон.

2. Кроме, того, что описано в п.1 на фото ничего нет.

3. Границы листа бумаги параллельны границам изображения.

4. Лист бумаги лежит на темном фоне, однородной текстуры.

5. Границы листа бумаги отчетливо (по всей длине, не прикрыты фоном, не выходят за границы фото) видны на изображении.

6. Лист бумаги строго формата A4.

7. Предмет не выходит за границы листа бумаги.

8. Необходимо отсутвие перспективного искажения.

**## Требования к массиву координат <a id ="pCoor"> </a>**

1. Программа работает с координатами в плоскости листа. Координаты измеряются в см, от левого нижнего края листа.

![Введенная СК.](/photos/Oxy/emptyOxy.jpg)

2. Количество координат изменяется в пределах [3, 16]. (Верхняя граница взята из воздуха, пока что ничем не осбусловлена. Чем она мб обусловлена? Временем работы алгоритма.)

3. Координаты лежат в промежутках:

    \* $x \in [0, 29.7]$

    \* $y \in [0, 21.0]$

4. Координаты вершины подаются в формате **\_\_'x, y'\_\_** разделены запятой и произвольным числом пробелов (ПЧП). Координаты вершин отделюят **\_\_';'\_\_** ПЧП. В конце массива координат стоит **\_\_'.'\_\_** ПЧП. (всё без ковычек).

//Пример

5, 5; 5, 7.9; 15.5, 7.9; 15.5, 5.

5. Направление обхода вершин по часовой стрелке

![Задали прямоугольник на введенной СК.](/photos/Oxy/example.jpg)

**## Сбор данных (План тестирования) <a id ="pData"></a>**

Имеем 10 предметов: (xLen, yLen) в см.

<ol>

 <li>Значок Музей Невская Застава (r = 3.8)</li>

 <li>Щипцы Jakemy (11.5 x 1.5)</li>

 <li>Зажигалка Volna (7.5 x 2.5)</li>

 <li>Торообразный тренажер (R = 8.8, r = 3.8)</li>

 <li>Флешка Raychem(2.1 x 4.8)</li>

 <li>Пилка Zinger(10.5 x 1.7)</li>

 <li>Болт для крепления GoPro (1.2 x 5.5)</li>

 <li>Тикер Токер «ИЗИ ИЗИ» (2.7 x 2)</li>

 <li>Бумажка в форме маечки (2.9 x 4)</li>

 <li>Маркер stabilo boss розовый (10.5 x 2.8)</li>

</ol>

Краевые условия для тестирования:

<ol>

 <li>Предметы [1, 10].</li>

 <li>Можно/Нельзя расположить одновременно.</li>

 <li>Прямоугольник/треугольник/... в качестве многоугольника.</li>

</ol>

Тесты:

<ol>

 <li>Один предмет, можно расположить. (Для каждого. 10 тестов.)</li>

 <li>Один предмет, нельзя расположить.

 (Просто менять. Для 10 дать 6, для 4 дать 1, для 8 дать 9. 3 теста.)</li>

 <li>Два предмета, можно расположить.

 (Точно дб тест с 4 и 1. 5 тестов.)</li>

 <li>Два предмета, нельзя расположить. (2 теста)</li>

 <li>Три предмета, можно расположить. (4 теста)</li>

 <li>Три предмета нельзя расположить. (2 теста)</li>

 <li>10 предметов, можно расположить. (1 тест)</li>

 <li>10 предметов, нельзя расположить. (1 тест)</li>

</ol>

Тотал: 28 тестов (для прямоугольников).

TODO: добавить 28 тестов для треугольников. + 28 тестов для произовльных четрыхугольников (выпуклых и невыпукалых).

**### Обоснование тестов:**

На ум приходит ММИ, то есть проверили для 1 для 2, а затем для n.

Но в этом случае рассмотреть три предмета считаю важным,

т.к. здесь уже точно определяешься с алгоритмом.

**## Усложнения <a id ="pHard"></a>**

\* Добавить возможность работать с криволинейными многоугольниками.

    \* Окружность, эллипс, кусочно заданные контуры.

**## Словарь обозначений <a id ="pDict"></a>**

\* <a id ="PCVM"> </a>ПЦВМ - Поместиться целиком в многоугольник значит, что предметы полностью находятся в выделенной области и не пересекают ее границы.

\* <a id ="3"> </a>(3): Вариант визуализации - обрисовываем контур предмета начиная с точки касающейся стороны многоугольника.

\* <a id ="OT"> </a>ОТ - особая точка [ЧЗОК](#ЧЗОК).

\* <a id ="43OK"> </a>ЧЗОК - число, род, падеж, склонение зависят от контекста.

\* <a id ="VIV"> </a> ВИВ - взято из воздуха, надо будет уточнить при реализации, скорее всего зависит от времени работы алгоритма.