Intelligent Placer

Требуется создать “Intelligent Placer”: по поданной на вход фотографии нескольких предметов на светлой горизонтальной поверхности и многоугольнику понимать, можно ли расположить одновременно все эти предметы на плоскости так, чтобы они влезли в этот многоугольник. Предметы и горизонтальная поверхность, которые могут оказаться на фотографии, заранее известны. Также заранее известно направление вертикальной оси Z у этих предметов. Многоугольник может задаваться 2-мя способами: а) массивом с координатами вершин на естественной плоскости (не в пикселях); б) фигурой, нарисованной темным маркером на белом листе бумаги, сфотографированной вместе с предметами. “Intelligent Placer” должен быть 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>])
```

которая возвращает **`True`** если предметы могут влезть в многоугольник, иначе **`False`.** То есть так, чтобы работал код:

```
from intelligent_placer_lib import intelligent_placer
def test_intelligent_placer():
    assert intelligent_placer.check_image(“/path/to/my/image.png”)
```

Также требуется воспроизводимый **`intelligent_placer.ipynb`**, содержащий репрезентативные примеры работы алгоритма с оценками качества его работы и их визуализацией.

Выше постановка задачи от заказчика (преподавателя). В ходе семестра требуется сформулировать свою постановку задачи; собрать под нее данные; продумать, реализовать и улучшить решение этой задачи так, чтобы оно удовлетворяло требуемому уровню качества; написать отчет о проделанной работе. Дальше все эти пункты описаны подробнее.

Вся проделанная работа от постановки задачи и данных до кода и отчета должна выкладываться в репозиторий на github. В корневом каталоге должны лежать, как минимум, **`intelligent_placer_lib`**, **`intelligent_placer.ipynb`** и **`README.md`** с постановкой задачи, планом алгоритма, проведенными улучшениями. В репозитории должны быть минимум 2 ветки: “main”(“master” - как стоит по-умолчанию) и “develop”. Все изменения в первую очередь заливаются в “develop”. К дедлайнам (23:59 в субботу после 1-й,4-й,7-й, и т.д. лекций) все изменения в “develop” надо будет коммитить в мастер и отправлять на ревью вашим коллегам. Коллегам будет даваться неделя на то, чтобы разобраться с изменениями и оставить комментарии с советами и отзывами о проделанной работе. В последнем ревью будет собираться итоговый отзыв.

Взаимодействие с заказчиком происходит по схеме:

1. Заказчик оценивает ваши намерения: постановку задачи, данные, план работы - смотрим, чтобы вы делали примерно то, что нужно.
2. Первые детальные оценки вашей работы с замечаниями и советами приходит от ваших коллег во время ревью.

3. Заказчику можно задавать вопросы на лекциях и семинарах; просить прокомментировать тот или иной фрагмент решения.
4. Вы предоставляете заказчику финальную версию вашего решения; он определяет его качество по прогону на своих данных; читает комментарии тех, что ревьюил ваше решение, анализирует некоторые детали и ставит итоговую оценку.

Постановка задачи

Требуется:

1. На свое усмотрение выбрать 10 небольших предметов, из которых будут выбираться предметы, попадающие на входную фотографию. Границы этих предметов должны четко выделяться на фоне белого листа бумаги. Пример набора предметов: ножницы, ручка, зеленая жвачка, монета, значок, мобильник, деталь из конструктора лего, перчатка, ботинок, зубная щетка. Важно, чтобы вы имели доступ к этим предметам на протяжении всего семестра (их придется неоднократно фотографировать).
2. Выбрать светлую горизонтальную поверхность, на фоне которой будут сниматься предметы на входной фотографии.
3. Сфотографировать сверху каждый из выбранных предметов на фоне белого листа бумаги A4 (который попадает на фотографию целиком, так чтобы углы и края листа были видны и не перекрывались). Сфотографировать сверху выбранную горизонтальную поверхность. Важно сделать этот шаг аккуратно, поскольку эти 11 фотографий надо будет передать нам и по ним будут автогенерироваться данные для оценки качества работы вашего решения.
4. Выбрать один из вариантов получения многоугольника: а) или б).
5. Четко сформулировать требования к входным и выходным данным в Вашей задаче: что Ваша программа может получать на вход и что она должна отдавать на выход. Виды требований:
    a. Фотометрические: допустимые разрешение фотографии, высота съемки, угол наклона камеры, степень размытости изображения, и другие
    b. По расположению объектов на фотографии: могут ли перекрывать друг друга и в какой степени, насколько далеко могут быть расположены от центра фотографии, толщина объектов, и другие
    c. Ограничивающие возможную форму многоугольника: например, число вершин не больше 10
    d. Может ли один объект присутствовать на фото несколько раз
6. Отправить сформулированные требования вместе с фотографиями предметов и поверхности заказчику. Получить его одобрение.
7. Выложить требования вместе с фотографиями в README.md в репозиторий.

Дедлайн на первую приемлемую версию: 23:59 в субботу после 1-й лекции. Базовая оценка: 4 балла.

Далее, в ходе решения задачи, сформулированные требования можно будет менять как в сторону усложнения, так и упрощения (до известных пределов).

Сбор данных

Требуется подготовить несколько десятков примеров входных данных для вашей задачи. Примеры должны быть максимально репрезентативны: покрывать как можно больше возможных типов конфигураций предметов и видов многоугольника.

Примеры необходимо разметить: сопоставить каждому из примеров ожидаемый результат работы алгоритма (поместятся ли предметы на нем в многоугольник или нет).

Далее ваша программа должна будет доставать эти размеченные примеры, запускать на них ваш алгоритм и сравнивать его результат с ожидаемым. Результат сравнения должен будет красиво выводиться (например, в таблицу), желательно с промежуточными результатами работы алгоритма (например, выделенными границами или особыми точками предметов).

Из всех примеров надо отобрать 10-15 наиболее репрезентативных, для каждого них написать, чем именно его случай примечателен. Эти примеры, а затем и результаты работы на них, нужно поместить на видное место в репозитории: по ним должно создаваться наиболее точное понимание, в каких случаях ваш алгоритм работает правильно, а в каких есть простор для улучшений.

Остальные примеры нужны для более детального понимания качества работы алгоритма и наиболее приоритетных направлений по его улучшению. Кроме того, при подборе примеров вы лучше поймете, как нужно решать поставленную задачу; какие в ней крайние случаи и подводные камни.

Понятно, что все случаи из 2^10 возможных наборов предметов с всевозможными видами многоугольников и всевозможными условиями съемки вы не переберете. Важно отобрать примеры, такие, что вероятность хорошей работы алгоритма на произвольном входе будет максимальна при условии хорошей работы алгоритма на отобранных примерах.

Дедлайн на первую приемлемую версию: 23:59 в субботу после 1-й лекции. Базовая оценка: 6 баллов.

Далее, в ходе решения задачи, собранные данные можно будет корректировать.

План решения задачи

Требуется написать: план решения задачи и согласовать его с заказчиком. Заказчик согласует план, если видит, что вы планируете двигаться в нужном направлении и шансы придти к алгоритму, решающему задачу с приемлемым качеством, высоки.

План должен содержать общую схему алгоритма и то, как вы планируете подбирать его параметры и улучшать. Алгоритм, как минимум, должен:

1. Находить признаки объектов на фотографии (их границы / особые точки / текстурные и визуальные признаки).
2. По найденным признакам идентифицировать объекты и понимать их размеры.
3. Оценивать, влезут ли оказавшиеся на фотографии предметы в многоугольник.

Дедлайн: 23:59 в субботу после 4-й лекции. Базовая оценка: 4 балла.

Реализация и итерации по улучшению

После одобрения постановки задачи, данных и плана, требуется реализовать решающий задачу алгоритм, а затем его улучшать. Для продумывания улучшений и их

приоритизации требуется оценивать качество работы решения. Полезно визуализировать работу решения на примерах разных типов.

Описание базового алгоритма, изменений в итерациях по улучшению, итоговый алгоритм и оценки его качества должны быть отражены в README.md.

Дедлайн: 23:59 в субботу, с которого начинается последнее ревью. В моменты отправок на промежуточные ревью в README.md должны оказываться актуальные на это время описания алгоритма и оценки его качества, возможно планы по дальнейшей работе.

Полученное решение оценивается по критериям

1. Разумность итогового алгоритма - возводится в 10 степень
    a. 0 баллов - решение отсутствует, не собирается, падает на исходных примерах
    b. 1 балл - алгоритм как-то решает поставленную задачу, но явно некачественно, особенности предметов и краевые случаи почти не учитывались
    c. 2 балла - алгоритм в целом решает задачу, но есть много легко реализуемых улучшений, которые существенно повышают качество работы
    d. 3 балла - алгоритм в целом решает задачу, легко реализуемых улучшений не много, но явно есть вещи, которые следовало бы написать иначе - и алгоритм работал бы намного качественнее
    e. 4 балла - алгоритм в целом решает задачу, идей как его точно можно существенно улучшить за разумное время нет, но есть много разумных гипотез про это, которые автор не проверил
    f. 5+ баллов - алгоритм в целом решает задачу, явных способов улучшить за разумное время не видно, автор проверил много гипотез
2. Качество работы полученного алгоритма на автосгенерированных нами данных (возможно отберет пару степеней у “Разумности”)

    TODO: пока под вопросом: зависит от того, насколько хорошо у нас получится автоматически проверять. Существенного влияния у этого пункта не будет.
3. Сложность решаемой задачи
    a. 1 балл - значительно проще базовой, расчет на слишком легкоразличимые предметы (например, исключительно по цветовым признакам) или очень простые многоугольники)
    b. 2 балла - решался базовый вариант
    c. 3 балла - есть минорные усложнения, “разумность алгоритма” на них не падает
    d. 4+ балла - есть существенные усложнения, примерно 2*(X+1) баллов, где X - число усложнений, разумность которых не ниже 4-х
4. Понятность описания алгоритма (оценивается сочетание словестного описания и кода)
    a. 0 баллов - невозможно разобраться даже с комментариями автора
    b. 1 балл - без комментариев автора разобраться невозможно
    c. 2+ балла - разобраться возможно (если человек явно старался - можно поставить 4-5 баллов)
5. Адекватность реагирования на замечания и советы
    a. 1 балл - на замечания не реагирует

    b. 5+ баллов - реагирует адекватно, разумные берет в работу, неразумные отклоняет, в случае непонимания и сомнений задает уточняющие вопросы
6. Разумность метрик качества, визуализация результатов работы решения
    a. 0 баллов - метрики качества отсутствуют
    b. 1 балл - метрики почти не коррелируют с качеством работы
    c. 5+ баллов - используются разумные метрики и визуализация, как их просто улучшить сходу непонятно.
7. Бонус за попадание в дедлайны. 2^X, где X - число раз, когда человек сдал итерацию до дедлайна (при условии итерация разумная: собирается + есть явные улучшения). Сдвиг может быть использован.

Итоговый отчет

В конце работы в README.md должны находиться:

1. Постановка задачи
2. Базовый алгоритм, итерации по его улучшению, итоговый алгоритм
3. Оценки качества результатов
4. Опционально: идеи по дальнейшему улучшению, особенности решения, интересные идеи по его применению и прочие комментарии

Дедлайн: 23:59 в субботу, с которой начинается последнее ревью. Базовая оценка: 4 балла. Случаи явного несоответствия отчета проделанной работе оцениваются в 0 баллов.

Ревью решений больших лабораторных своих коллег

Раз в 3 недели каждому из вас надо будет провести ревью работ других студентов: получить от них ссылку на коммит с изменениями, разобраться в них и оставить в комментариях советы с замечаниями, некоторые моменты возможно обсудить (также в комментариях). При ревью требуется обращать внимание именно на алгоритмическую составляющую решения, стиль кодирования второстепенен.

Автор - тот, чью работу ревьюят.
Ревьюер (reviewer) - тот, кто ревьюит.

По-умолчанию на ревью всегда выделяется неделя, при возникновении дискуссий дедлайн может быть сдвинут еще на неделю при согласии обоих участников.

В первом ревью каждый студент получает случайно выбранные 8 постановок задач и данных других студентов и ревьюит их. После этого студенты могут отправить нам пожелания, кого они хотят либо не хотят видеть в числе своих ревьюеров и авторов. С учетом этого мы распределяем работы, так чтобы каждому досталось по 4 работы на ревью и по 4 ревьюера для своей работы. Далее авторы добавляют вас в число контрибьюторов проекта и это деление сохраняется до конца семестра.

Итоги работы ревьюеров оцениваются по критериям:

1. Разумность оставляемых комментариев.
    a. 0 баллов - комментарии практически отсутствуют; все что есть, это странные придирки к коду и прочие, не имеющие отношения к алгоритму

    b. 5+ баллов - комментарии присутствуют, по ним видно, что человек хорошо разобрался в работе; где надо - похвалил хорошее решение; где надо - оставил понятное замечание; где надо - разумно отвечал на вопросы автора + написал разумный отзыв. Если таких очень хороших ревью много - умножать на их количество.
2. Число работ, в которых ревьюер разобрался и оставил осмысленные советы либо отзывы. Число баллов примерно равно числу таких работ (варьируется в зависимости от качества).
3. Качество формулировок в замечаниях
    a. 1 балл - без доп вопросов сложно понять, что человек имеет в виду
    b. 2 балл - комментарии понятные, но содержат слишком много текста, явно можно сформулировать короче и понятнее
    c. 3+ балла - комментарии понятные и лаконичные
4. Пунктуальность
    a. 1 балл - ревьюер оставлял свои комментарии в конце дедлайнов либо после них, отвечал на вопросы поздно, возможности обсудить почти не было
    b. 4 балла - ревьюер оставлял комментарии вначале отведенного на ревью срока, оставалось достаточно времени, чтобы на них ответить и их обсудить