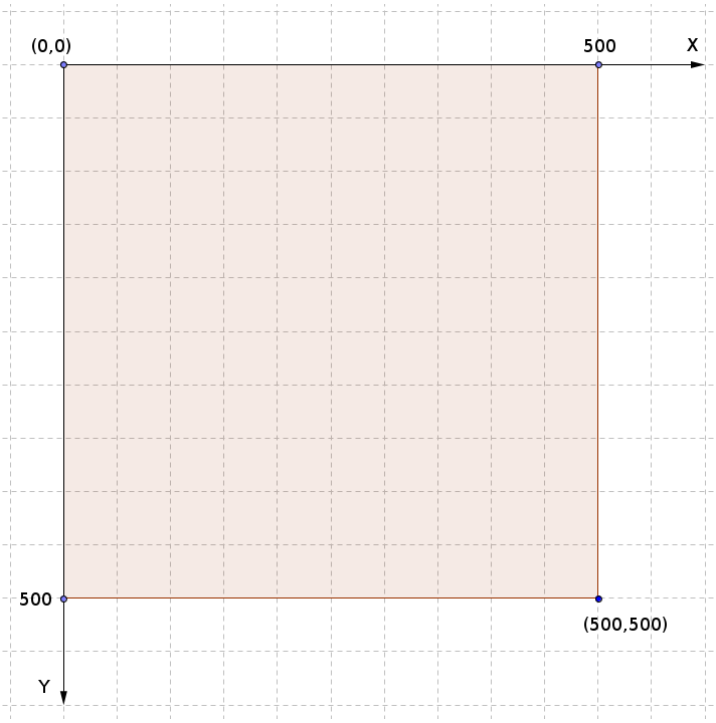
Школа компьютерного зрения 3DIVI (осень 2017)
Тестовое задание

1) Техническая часть

Реализовать генератор серых 8-битных изображений размера 500x500 пикселей в соответствии со следующим алгоритмом («серый 8-битный» означает, что значение каждого пикселя – целое число от 0 до 255).

Координатная система: левый верхний угол изображения имеет координаты [0; 0], правый нижний угол изображения – [500; 500] (при этом индекс правого нижнего пикселя – [499;499]).



Система координат

-Случайно сгенерировать длину стороны куба равномерно из диапазона [150; 300] пикселей (длина может быть нецелой).

-Случайно сгенерировать матрицу вращения, как произведение матриц вращения вокруг осей X, Y, Z . Три угла выбираются случайно равномерно из интервала [0; 2π].
https://ru.wikipedia.org/wiki/Матрица_поворота

-Сгенерировать случайно и равномерно точку (x, y) на изображении.

-Применить вращение к кубу, перенести его так, чтобы центр куба был в точке (x, y) и ортогонально спроецировать на изображение.

-Повторить генерацию заново, если:

1) какая-либо вершина куба не попала на изображение;

2) минимальное расстояние между всеми парами вершин оказалось менее 100;

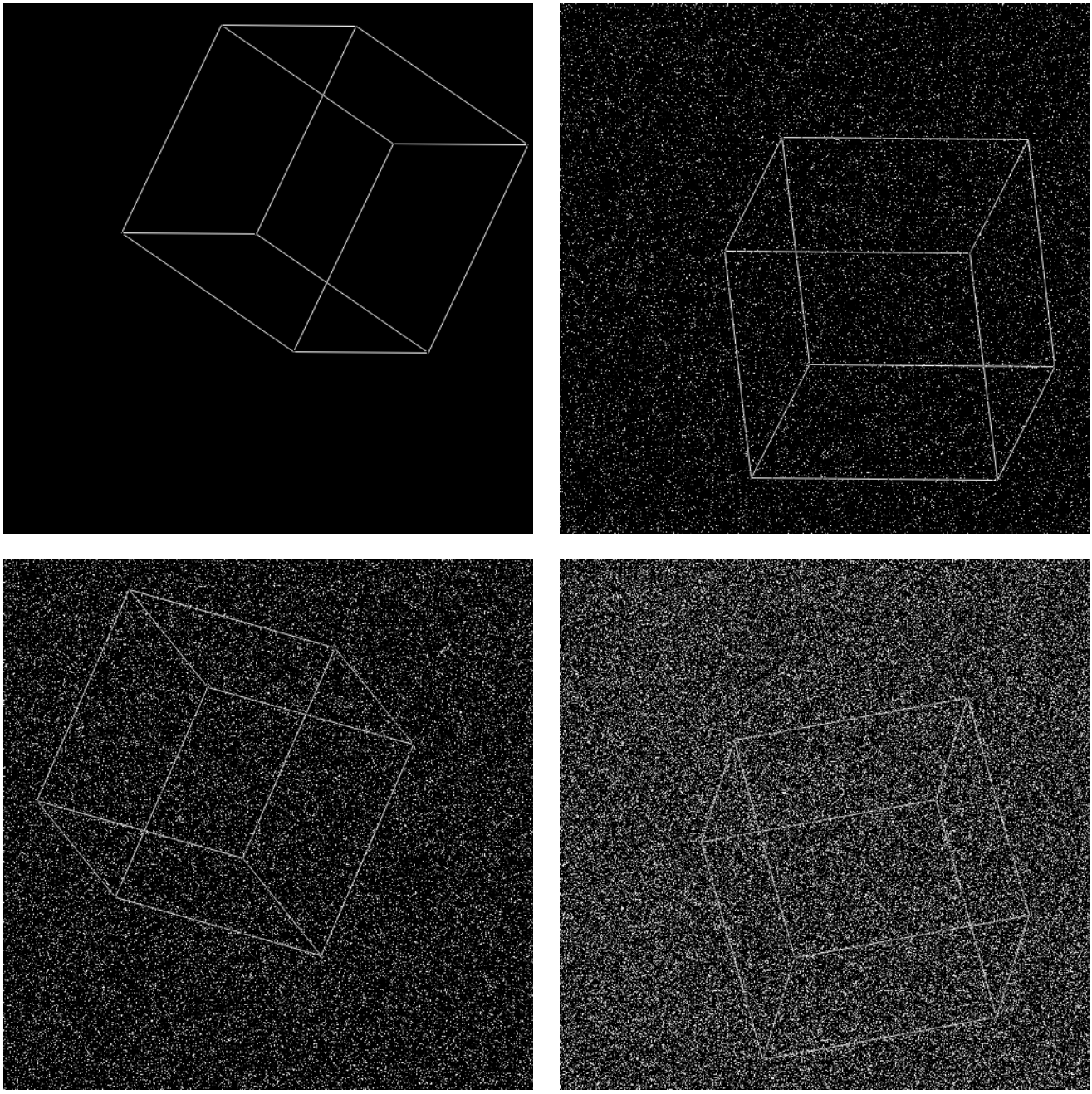
3) минимальное попарное расстояние для хотя бы одной из трёх четвёрок параллельных прямых, содержащих ребра куба оказалось менее 50.

-Нарисовать 12 ребер куба с помощью алгоритма *Wu*:

https://en.wikipedia.org/wiki/Xiaolin_Wu's_line_algorithm

(необходимо реализовать алгоритм).

- Произвести зашумление: значение каждого пикселя с вероятностью P заменяется на случайное целое, равномерно распределенное на отрезке [0; 255].



Примеры сгенерированных изображений для P = 0; 0.1; 0.2; 0.4.

2) Алгоритмическая часть

Придумать и реализовать алгоритм восстановления координат вершин куба по изображению, полученному из генератора.

Требования к решению

1. Решение должно быть написано на одном из языков C++, Python.
2. Ограничение на время работы на одном тесте – 1 секунда.
3. Синтаксис вызова программы с параметрами запуска (командной строки):

а) для генерации изображения:

solver -generate P

где P – вероятность зашумления пикселей. После запуска в текущей директории должен появляться файл image.pgm со сгенерированным изображением. Изображение должно быть сохранено в формате PGM (https://en.wikipedia.org/wiki/Netpbm_format) - простой «текстовый» формат изображений.

б) для восстановления вершин:

solver -restore image.pgm

После запуска в файл output.txt должны сохраняться координаты восьми вершин куба в произвольном порядке. Координаты (x, y) каждой вершины должны находиться на отдельной строке. Пример output.txt:

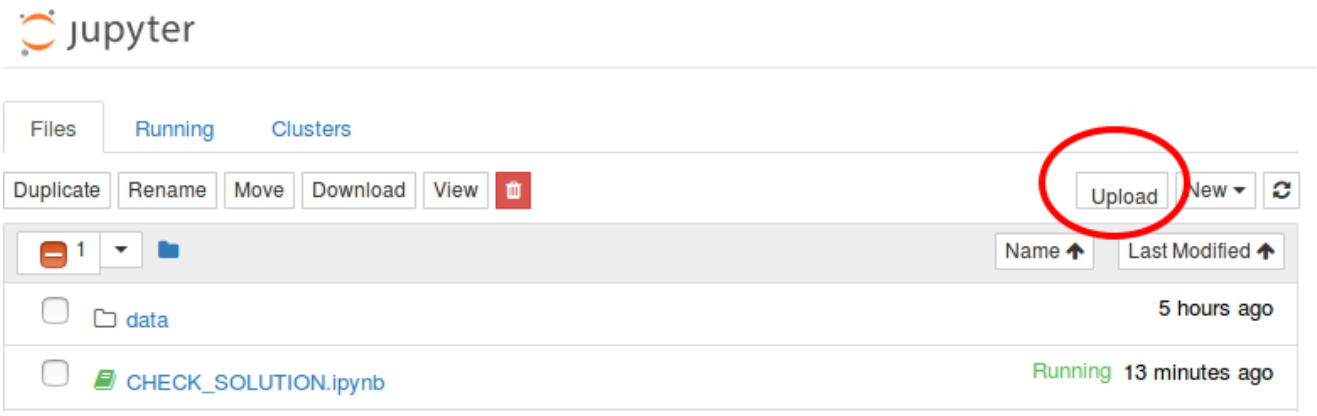
273.818 328.447
111.829 216.340
238.704 217.783
400.693 329.889
367.296 132.217
205.307 20.111
332.182 21.553
494.171 133.660

4. Решение будет проверяться на закрытом наборе тестов различной сложности (различные значения параметра P). Для каждого теста будет оцениваться максимальное отклонение по 8-ми вершинам (мера отклонения - евклидово расстояние до правильной позиции вершины). Если максимальное из 8-ми отклонений больше 5, то максимальное отклонение для данного теста полагается равным 5. За итоговую оценку решения будет приниматься среднее максимальное отклонение по всем тестам.
5. Все файлы решения (файлы исходного кода, заголовочные файлы, скрипты) должны быть в одной папке (без вложенных папок).
6. Для C++, при проверке, решение будет собираться командой:

g++ -o solver --std=c++11 *.cpp

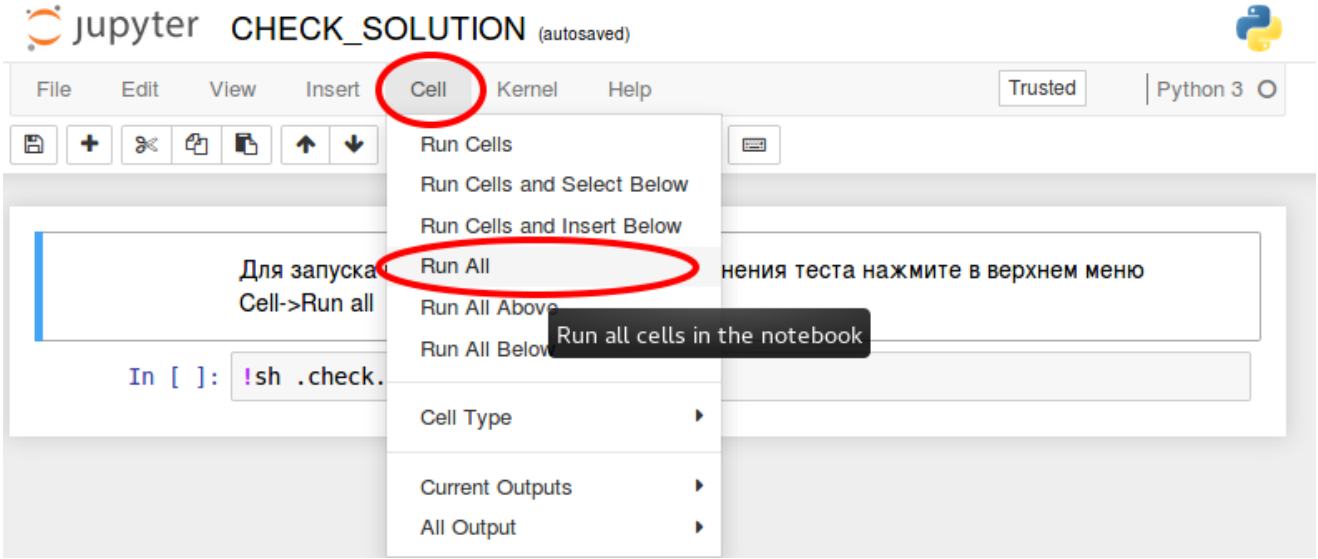
7. Перед отправкой необходимо проверить решение на совместимость с системой проверки (компилируемость, использование библиотек, верный формат вывода).

Для этого нужно перейти на сайт:
http://school.3divi.com:8000/
Откроется временный jupyter notebook.
Сделать *Upload* всех файлов исходного кода.



Внимание: *это временный сервер, время его жизни — 30 минут. Он будет удален со всеми загруженными файлами. Также он будет удален в случае неактивности в течение 5-ти минут.*

8. Для запуска проверки открыть файл *CHECK_SOLUTION.ipynb.*
9. В пункте меню *Cell* нажать *Run all*.
В случае неудачи будут выведены ошибки.



Решение будет проверено на открытом наборе тестов (изображения и ответы лежат в папке *data*, название папки – вероятность шума P).

10. Если проверка пройдена, можно отправлять итоговое решение на почту с указанием полученного значения **Total error**.
Окончательная проверка будет проводится на закрытом наборе данных.

Целью выполнения задания является проверка базовых навыков программирования и решения алгоритмических задач, поэтому при решении не предполагается использование дополнительных библиотек (для C++ естественно допускается использовать STL).