Analyzing results from XML files

Requirements

Based on the sample data present in this zip file, write a program that would produce a text file with the number of Ground Control Points (GCPs), the approximate area covered by them, and for each of the GCPs in how many images it can be seen.

Your program should be run from the command line:

```
my_program.py input.p4d output.txt
```

An example output would then look like this:

```
3
1500.37
0: 7
1: 10
2: 9
```

3 is the number of GCPs in the file, 1500.37 the approximate area they cover and 7, 10, 9 the number of images that see GCP 0, 1, 2 respectively.

> Read carefully the description with the details and hints following below.

Explanations

P4D file format

Evaluating Pix4D software relies a lot on extracting useful data from input/output/log files.
Some of these files are structured in an XML fashion.
This is the case of the .p4d file format that describes projects (see the accompanying example data).

Ground Control Points (GCPs)

Ground Control Points (GCPs) are reference 3D points whose position is known accurately and that are used to bind photogrammetry projects to well known coordinate systems.

In the .p4d file passed as input, search for the `<GCP>` tags enclosed by a `<gcps>` tag.
Each GCP has (among other properties) a associated unique id `<id>` and x and y coordinates which are what we are interested in.
You can discard the other values (altitude, longitude…).

The .p4d files also contain information about the images in the projects enclosed by `<image>` tags inside an `<images>` section.
Inside `<imageGCP>` tags there are references to the GCPs which are seen in this image, using the same IDs as described before.

Approximate area

We just want to compute a crude approximation of the area covered by the GCPs.
Thus, we will:

- discard the altitude / z-coordinate and assume all the points are in the same plane
- replace the shape described by teh GCPs by their axis-aligned bounding box.

For example, if we have 3 GCPs with `(x,y)`-coordinates `(0, 0)`, `(10, 0)`, `(0,11)`, we will consider the rectangle defined by the 4 corners `(0,0)`, `(10,0)`, `(10,11)`, `(0,11)` and the area we want to compute is the area of this rectangle (110 in this case).

Recommandations

- The main focus of this exercise is the correctness and clarity of the implementation.
- Do not hesitate to contact us if something is not clear!
- Use Python 3.5 at least.
- You're free to use any module that you find useful.