Project 6 (Atomic Nature of Matter)

Clarifications and Hints

Prologue

Project goal: re-affirm the atomic nature of matter by tracking the motion of particles undergoing Brownian motion, fitting this data to Einstein's model, and estimating Avogadro's number

The zip file (`http://www.swamiiyer.net/cs110/atomic_nature_of_matter.zip`) for the project contains

- project specification (`atomic_nature_of_matter.pdf`)
- starter files
  - `blob.py`
  - `blob_finder.py`
  - `bead_tracker.py`
  - `avogadro.py`
- test script (`run_tests.py`)
- test data (`data/`)
- report template (`report.txt`)

This checklist will help only if you have read the writeup for the project and have a good understanding of the problems involved. So, please read the project writeup before you continue with this checklist.

Problems

Problem 1 (*Particle Identification*) Define a data type `Blob` that has the following API:

method	description
`Blob()`	an empty blob b
`b.add(i, j)`	add a pixel (i, j) to the b
`b.mass()`	the number of pixels in b, ie, its mass
`b.distanceTo(c)`	the distance between the centers of b and c
`str(b)`	string representation of b's mass and center of mass

Next, define a data type `BlobFinder` that has the following API:

method	description
`BlobFinder(pic, tau)`	a blob finder bf to find blobs in the picture pic using a luminance threshold tau
`bf.getBeads(P)`	list of all beads with $\geq P$ pixels

Hints

- `Blob`
  - Instance variables
    - Number of pixels, `_P` (`int`)
    - x-coordinate of center of mass, `_x` (`float`)
    - y-coordinate of center of mass, `_y` (`float`)
  - `Blob()`
    - Initialize the instance variables appropriately
  - `b.add(i, j)`
    - Use the idea of *running average*[1] to update the x- and y-coordinates of the center of mass of blob `b` to include the new point (`i`, `j`)
    - Increment the number of pixels in blob `b` by 1
  - `b.mass()`
    - Return the number of pixels in the blob `b`
  - `b.distanceTo(c)`
    - Return the Euclidean distance between the center of mass of blob `b` and the center of mass of blob `c`

[1] If $\bar{x}_{n-1}$ is the average value of $n-1$ points $x_1, x_2, \ldots, x_{n-1}$, then the average value $\bar{x}_n$ of n points $x_1, x_2, \ldots, x_{n-1}, x_n$ is $\frac{\bar{x}_{n-1} \cdot x_n}{n}$

Problems

- `BlobFinder`
  - Instance variable
    - Blobs identified by this blob finder, `_blobs` (`list` of `Blob` objects)
  - `BlobFinder()`
    - Initialize `_blobs` to an empty list
    - Create a 2D list of booleans called `marked`, having the same dimensions as `pic`
    - Enumerate the pixels of `pic`, and for each pixel `(i, j)`: 1. Create a `Blob` object called `blob`; 2. Call `_findBlob()` with the appropriate arguments; and 3. Add `blob` to `_blobs` if it has a non-zero mass
  - `bf._findBlob()`
    - Base case: return if pixel `(i, j)` is out of bounds, or if it is marked, or if its luminance (use the function `luminance.luminance()` for this) is less than `tau`
    - Mark the pixel `(i, j)`
    - Add the pixel `(i, j)` to the blob `blob`
    - Recursively call `_findBlob()` on the N, E, W, and S pixels
  - `bf.getBeads(P)`
    - Return a list of blobs from `_blobs` that have a mass of at least `P`

Problem 2 (*Particle Tracking*) Implement a client program `bead_tracker.py` that takes an integer P, a float tau, a float $delta$, and a sequence of JPEG filenames as command-line arguments, identifies the beads in each JPEG image using `BlobFinder`, and prints out (one per line, formatted with 4 decimal places to the right of decimal point) the radial distance that each bead moves from one frame to the next (assuming it is no more than $delta$).

Hints

- Read command-line arguments `P`, `tau`, and `delta`
- Construct a `BlobFinder` object for the frame `sys.argv[4]` and from it get a list of beads `prevBeads` that have at least `P` pixels
- For each frame starting at `sys.argv[5]`, construct a `BlobFinder` object and from it get a list of beads `currBeads` that have at least `P` pixels
- For each bead `currBead` in `currBeads`, find a bead `prevBead` from `prevBeads` that is no further than `delta` and is closest to `currBead`, and if such a bead is found, write its distance (using format string `'%.4f\n'`) to `currBead`
- Write a newline character
- Set `prevBeads` to `currBeads`

Problem 3 (*Data Analysis*) Implement a client program `avogadro.py` that reads in the displacements from standard input and computes an estimate of Boltzmann's constant and Avogadro's number using the formulae described above.

Hints

- Calculate `var` as the sum of the squares of the `n` displacements (each converted from pixels to meters) read from standard input
- Divide `var` by `2 * n`
- Initialize `eta`, `rho`, `T`, and `R` to appropriate values
- Estimate Boltzman constant `k` as `6 * math.pi * var * eta * rho / T`
- Estimate Avogadro's number `N_A` as `R / k`
- Write `k` and `N_A` using format string `'%e'` (for scientific notation)

Epilogue

Your project report (use the given template, `report.txt`) must include

- time (in hours) spent on the project
- short description of how you approached each problem, issues you encountered, and how you resolved those issues
- acknowledgement of any help you received
- other comments (what you learned from the project, whether or not you enjoyed working on it, etc.)

Before you submit your files

- make sure your programs meet the input and output specifications by running the following command on the terminal

```
$ python run_tests.py -v [<problems>]
```

- make sure your programs meet the style requirements by running the following command on the terminal

```
$ pep8 <program>
```

- make sure your report isn't too verbose, doesn't contain lines that exceed 80 characters, and doesn't contain spelling/grammatical mistakes