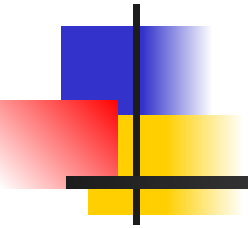


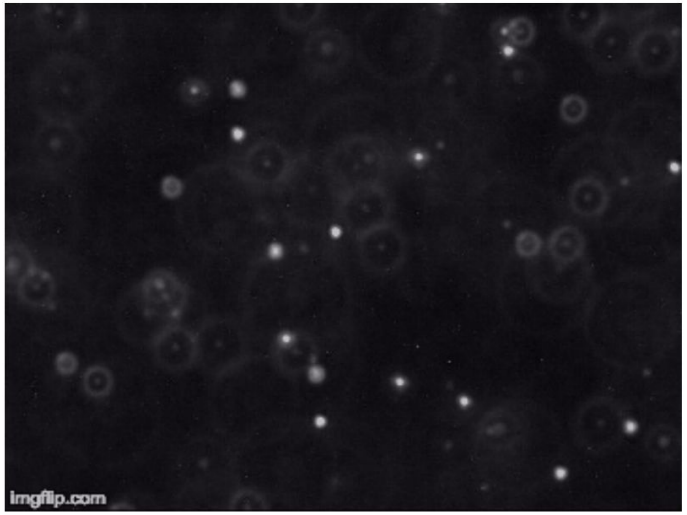
COS 126 – Atomic Theory of Matter

Questions

Raise your hand and ask

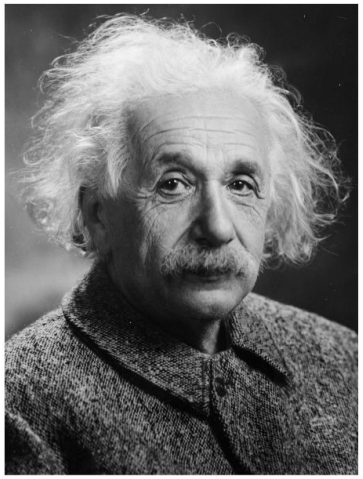
Ask on Ed

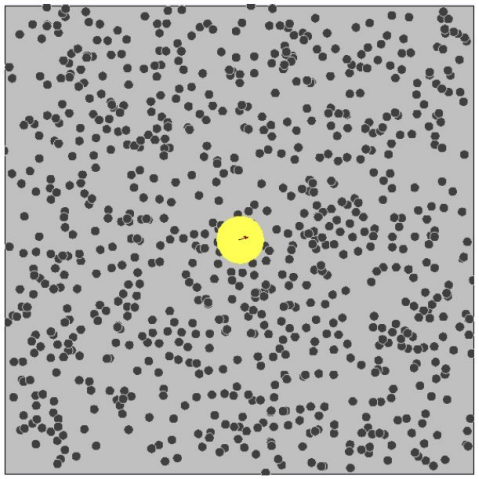
Project Context: 1827



- No universal acceptance of the atomic nature of matter
- Botanist Robert Brown notices erratic motion of pollen grains in water. This motion is later called: *Brownian motion*

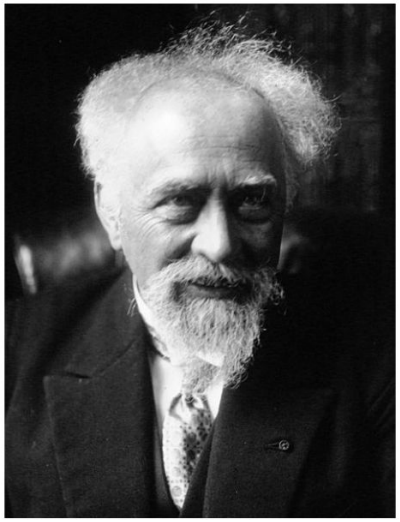
Project Context: 1905

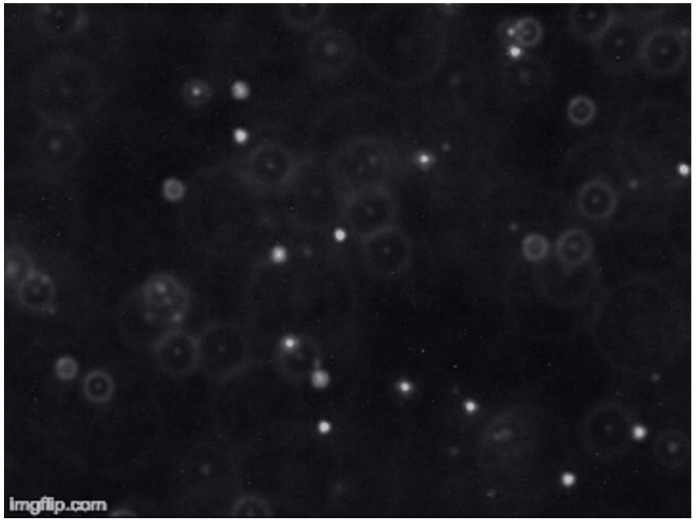




- Einstein publishes a revolutionary paper:
- Brownian motion is caused by smaller moving particles colliding with the larger pollen grains.
- Density of particles affects displacement in Brownian motion

Project Context: 1908





- Jean Baptist Perrin experimentally validated Einstein's theory and equations.

Project Context: 2023

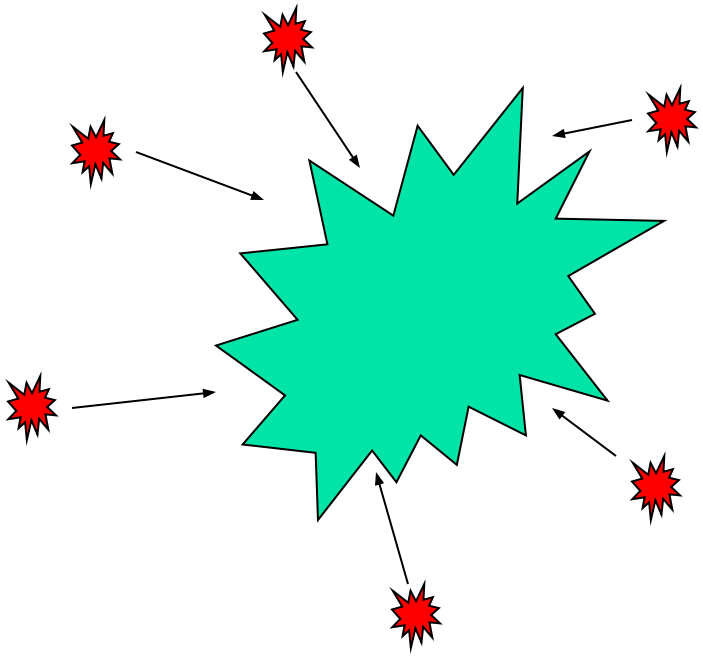
- Your Task: Redo Perrin's experiments!
- Not so difficult with computers and your COS126 skills

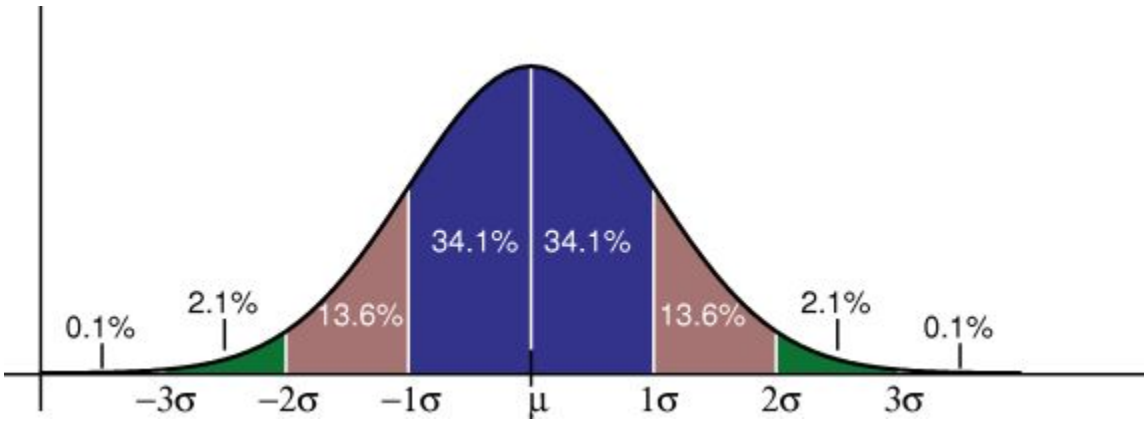
Goal of the Assignment

- Calculate Avogadro's number
  - Using Einstein's equations
  - Using fluorescent imaging
- Input data
  - Sequence of images
  - Each image is a rectangle of pixels
  - Each pixel is either light or dark
- Output
  - Estimate of Avogadro's number

Atomic Theory Overview

- Brownian Motion
  - Random collision of molecules
  - Displacement over time fits a Gaussian distribution

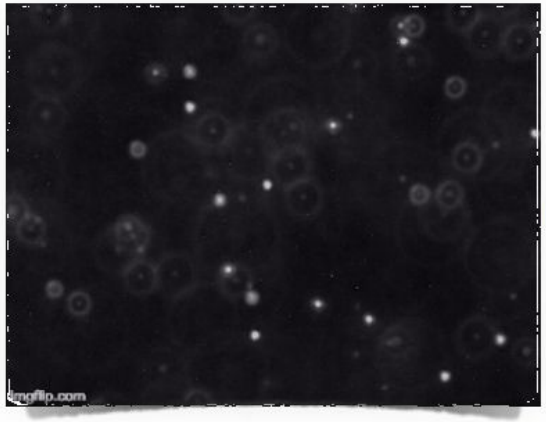




Atomic Theory Overview

- Avogadro's Number
  - Number of atoms needed to equal substance's atomic mass in grams
    - N_A atoms of Carbon-12 = 12 grams
  - Can calculate from Brownian Motion
    - Variance of Gaussian distribution is a function of resistance in water, number of molecules

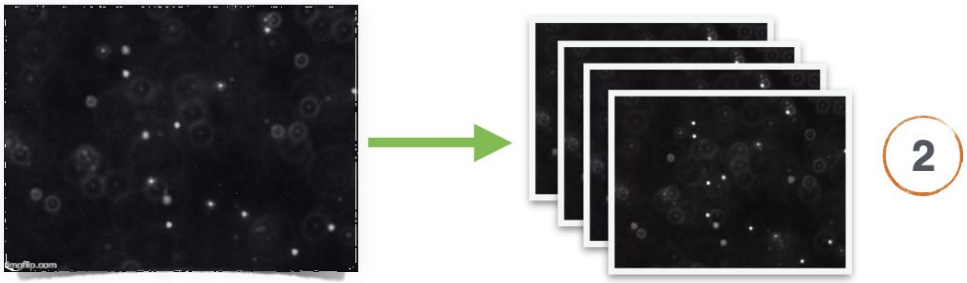
Experiment Overview



1

Record a video of particles undergoing Brownian motion

Experiment Overview

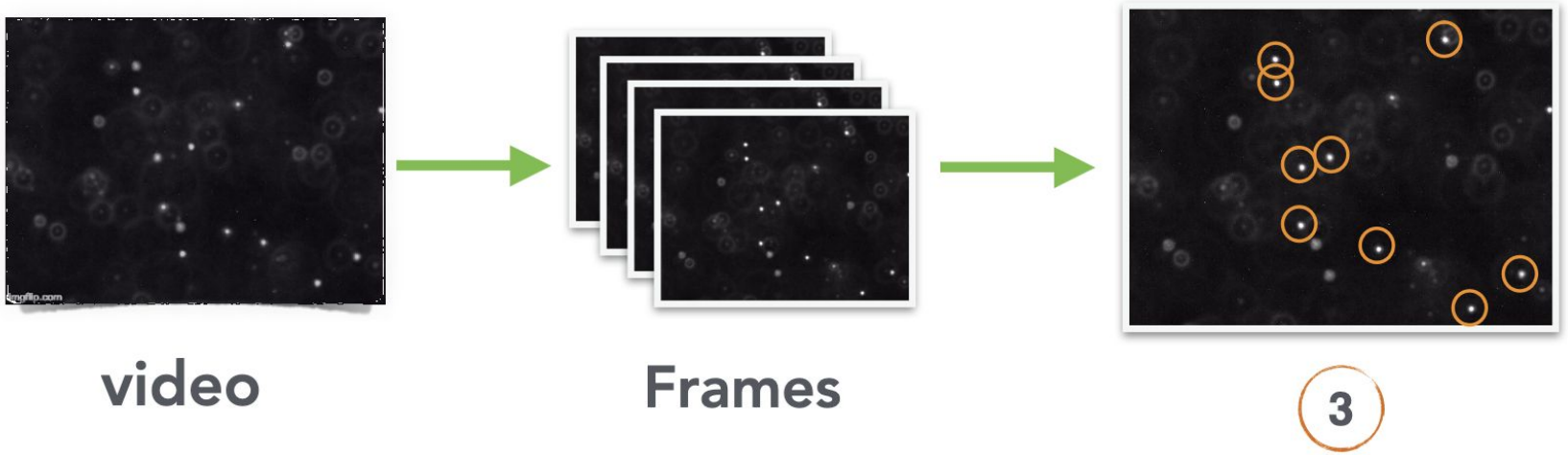


video

2

Convert the video into a sequence of frames

Experiment Overview

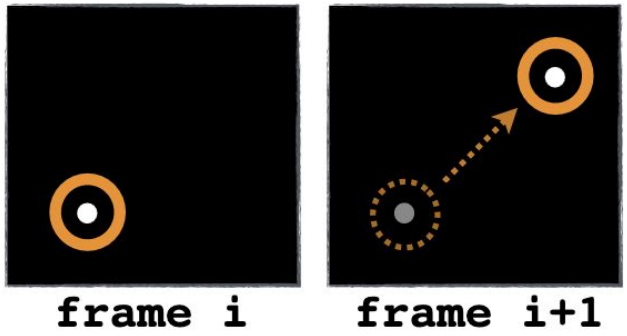


Identify ***Beads*** in every frame

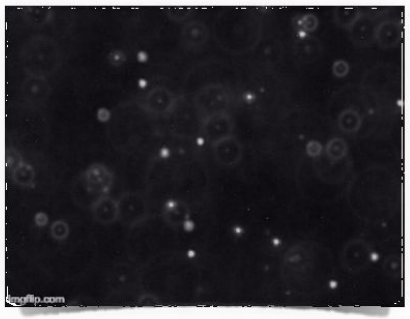
Experiment Overview

4

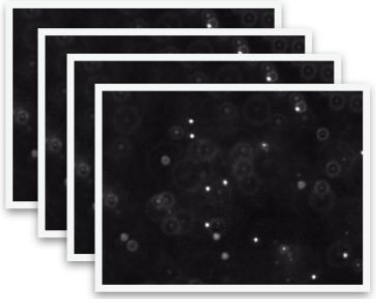
Compare positions of beads in every two consecutive frames



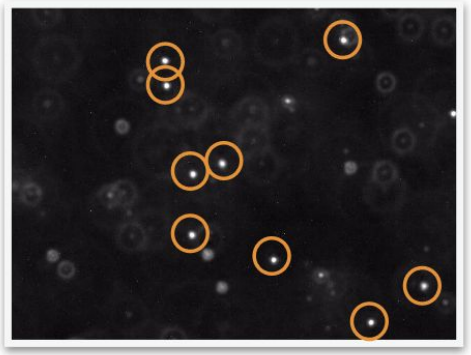
Experiment Overview



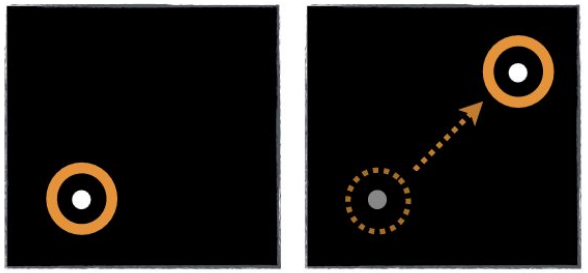
video



Frames



Beads

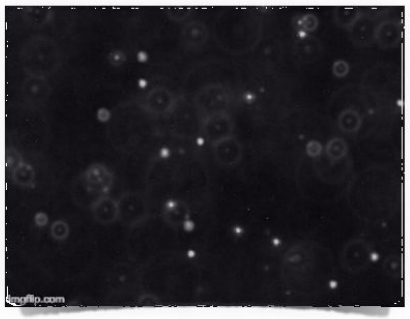


7.1833
4.7932
2.1693
5.5287
5.4292
2.1893
5.7294
3.1141
4.5576
1.9898
.....

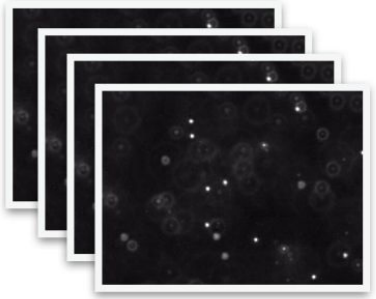
5

Record all bead **displacements** across consecutive fames.

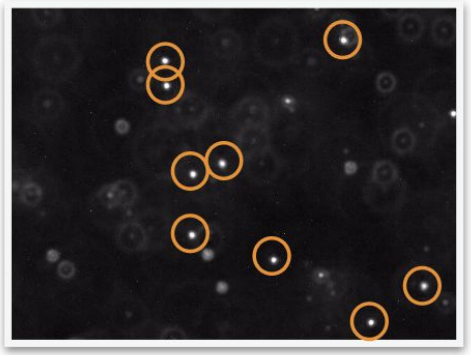
Experiment Overview



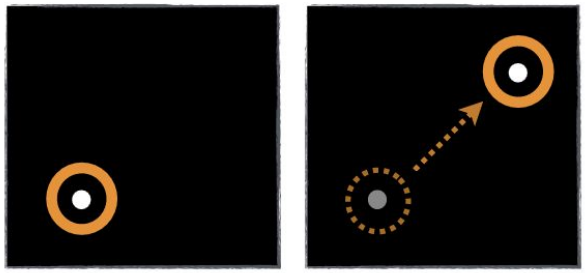
video



Frames



Beads

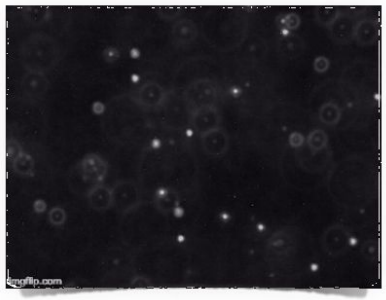


7.1833
4.7932
2.1693
5.5287
5.4292
2.1893
5.7294
3.1141
4.5576
1.9898
.....

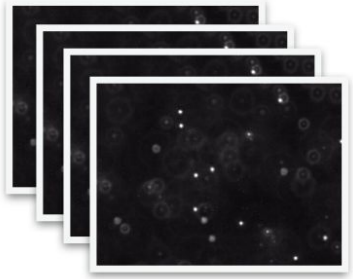
5

Record all bead **displacements** across consecutive fames.

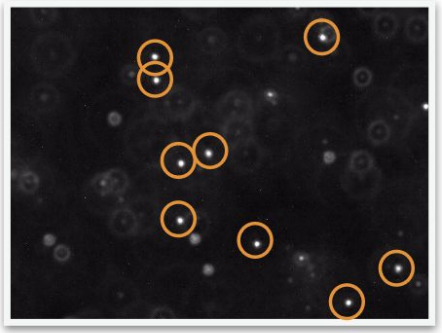
Experiment Overview



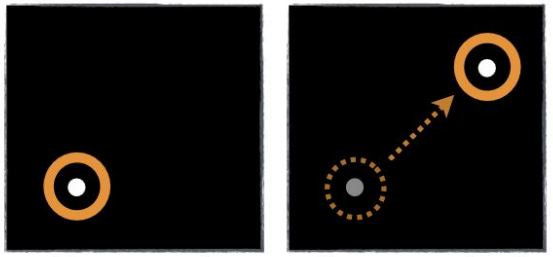
video



Frames



Beads

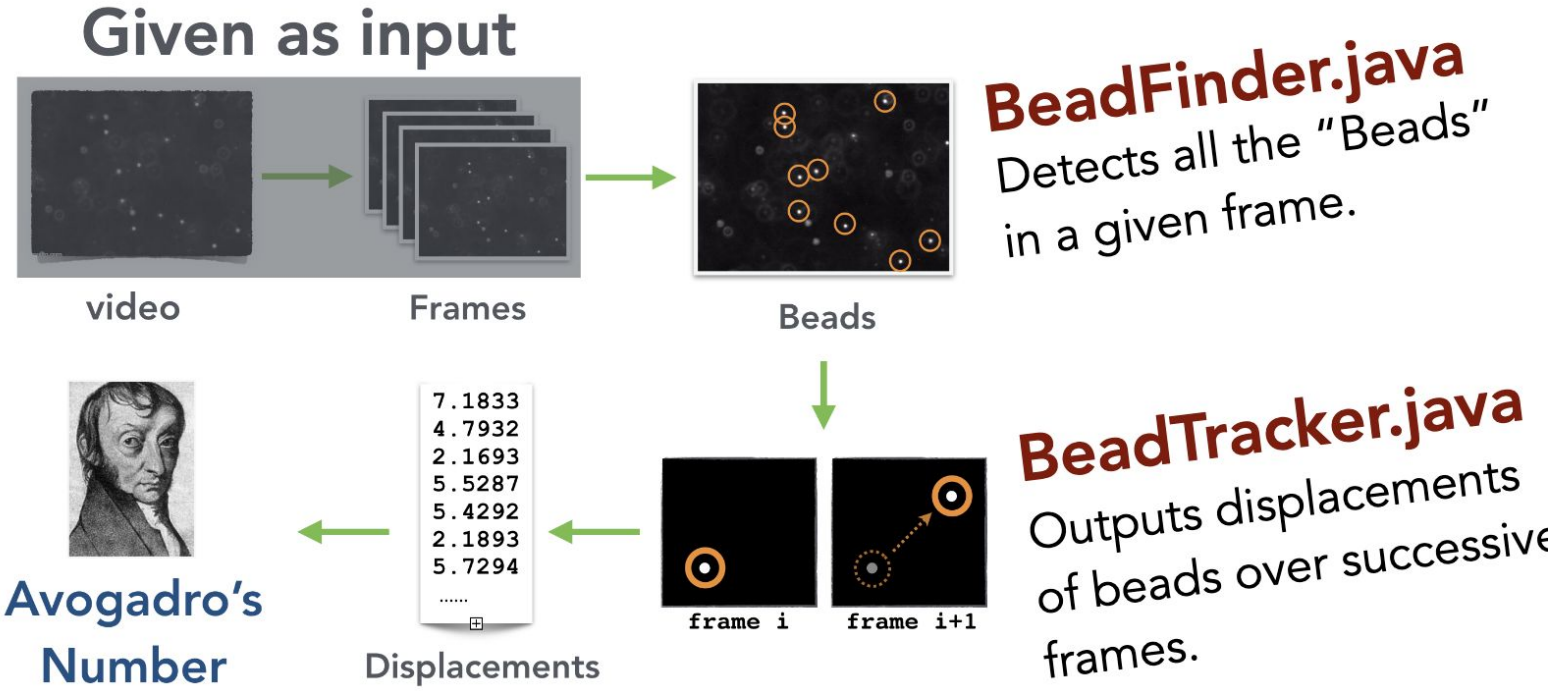


7.1833
4.7932
2.1693
5.5287
5.4292
2.1893
5.7294
.....

Displacements

Avogadro's Number

Experiment Overview



Avogadro.java
Computes Avogadro's number from a given set of displacements.

+ **Blob.java** Represents a set of adjacent pixels.

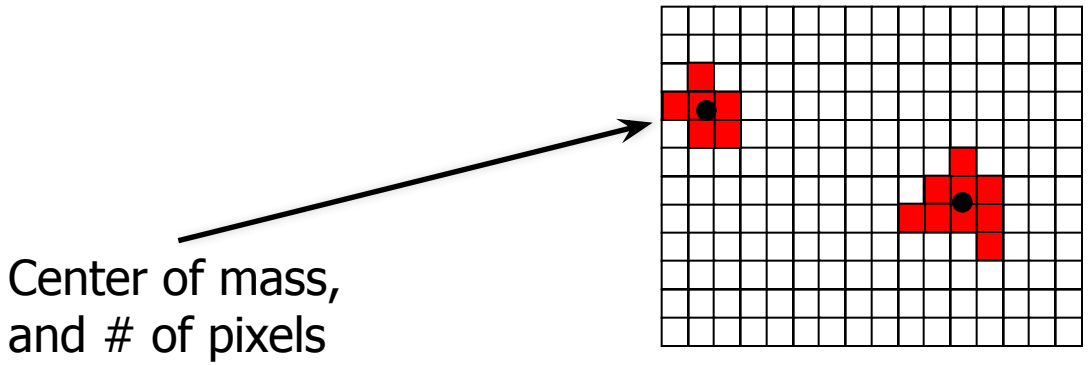
+ **readme.txt** Shows performance analysis.

Assignment: Four Programs

- Blob.java
  - Represents a set of adjacent pixels.
- BeadFinder.java
  - Detects all the *Beads* in a given image.
- BeadTracker.java
  - Outputs displacements of beads over consecutive frames.
- Avogadro.java
  - Computes Avogadro's number from a given set of displacements.
- readme.txt
  - Shows performance analysis.

Blob.java

- API for representing particles (blobs) in water
  - `public Blob()`
  - `public void add(int i, int j)`
  - `public int mass()  // number of pixels`
  - `public double distanceTo(Blob b) // from center (average)`
  - `public String toString()`
- Only need *three* values to efficiently store
  - Do *not* store the positions of every pixel in the blob

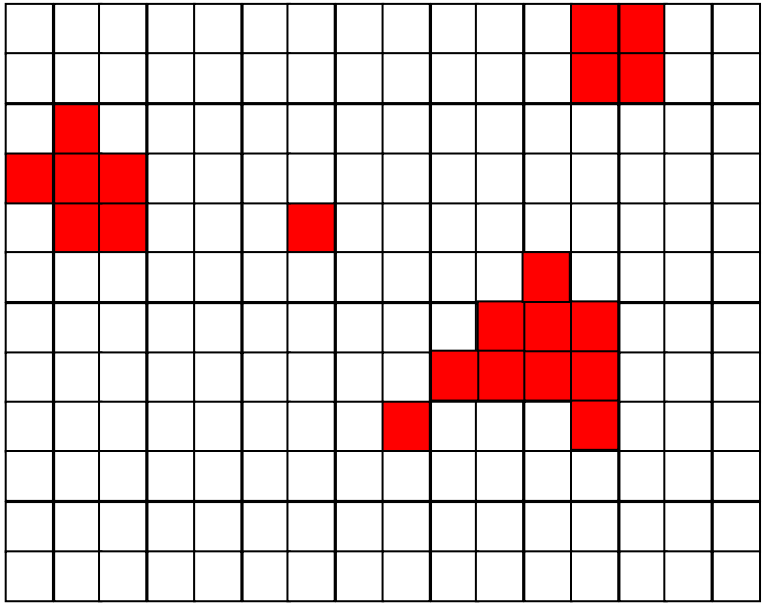


Blob Challenges

- Format numbers in a nice way
  - `String.format("%2d (%8.4f, %8.4f)", mass, cx, cy);`
  - (Use same format in `System.out.printf()`)
  - E.g., `"%6.3f" -> _2.354`
  - E.g., `"%10.4e" -> 1.2535e-23`
- Thoroughly test
  - Create a simple `main()`

Blobs and Beads

- Blob: Any group of adjacent light pixels.
  - Not 
- How many blobs are there?
- Bead: A blob with a number of pixels that is at least min.
- How many beads are there? (assume min=5)



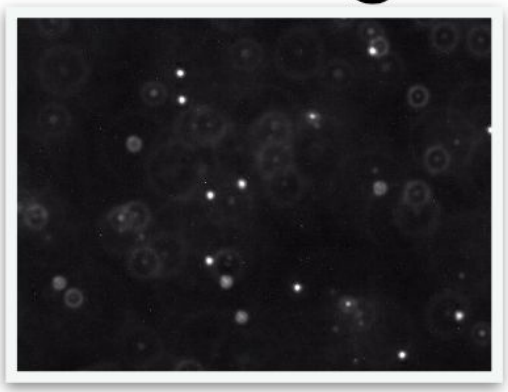
BeadFinder.java

- Locate all blobs in a given image
  - And identify large blobs (called beads)
- API
  - public BeadFinder(Picture picture, double threshold)
    - Calculate luminance (see Luminance.java, 3.1)
      - Include pixels with a luminance >= threshold
    - Find blobs with DFS (see Percolation.java, 2.4)
      - The hard part, next slide…
  - public Blob[] getBeads(int minSize)
    - Returns all beads with at least minSize pixels
    - Array must be of size equal to number of beads

BeadFinder.java

Input:

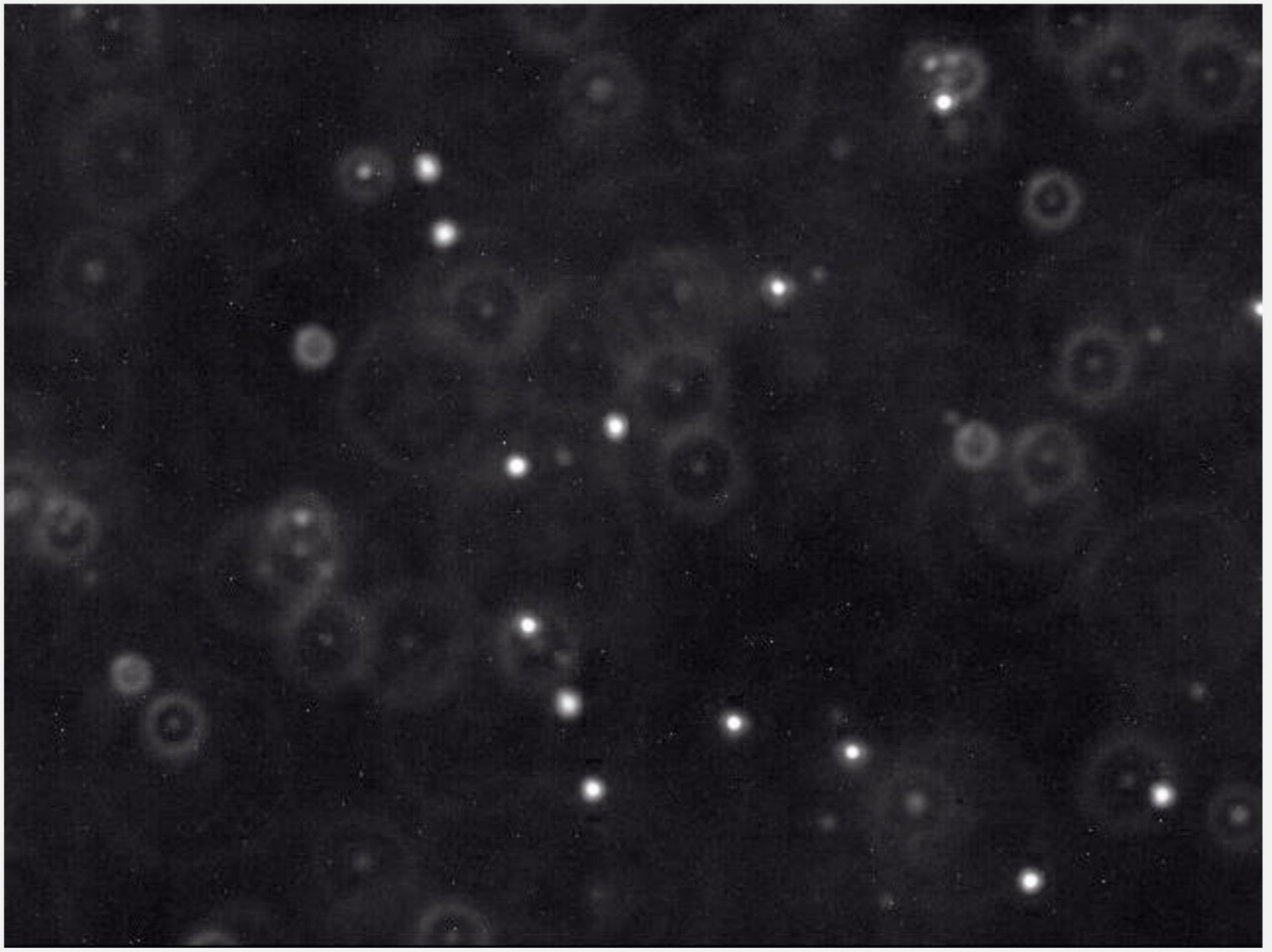
An Image



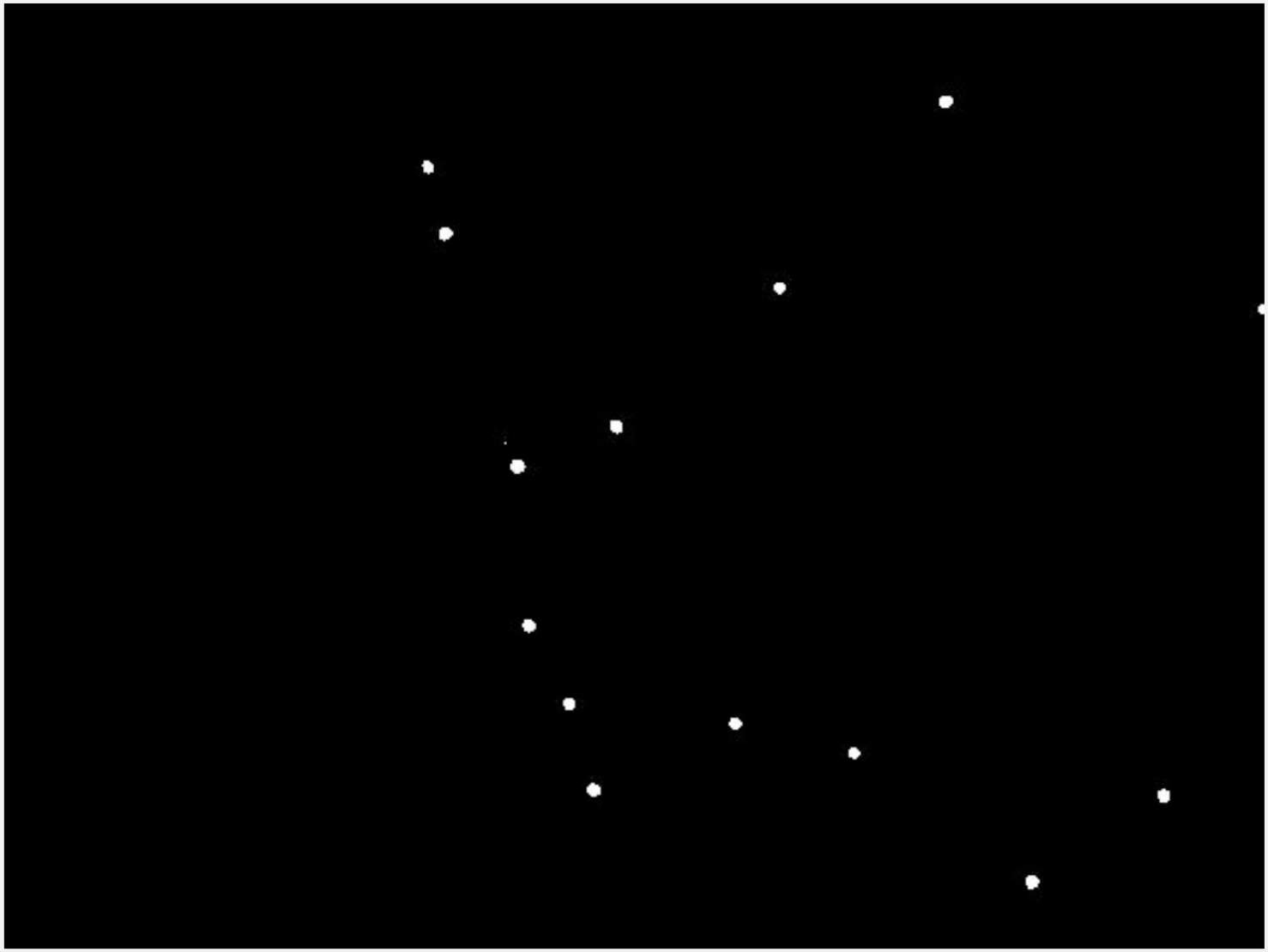
A Luminance Threshold ***tau***

BeadFinder.java

BeadFinder: Original Image

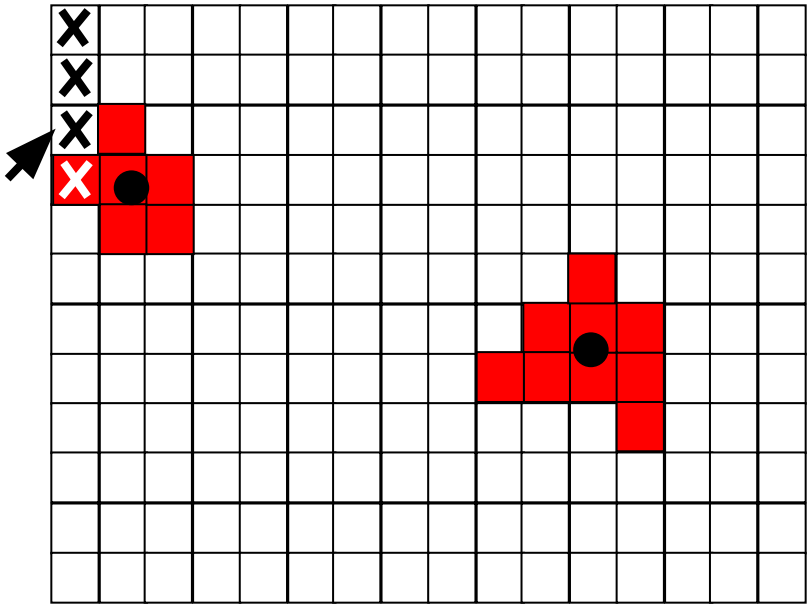


BeadFinder: Applying Luminance Threshold tau



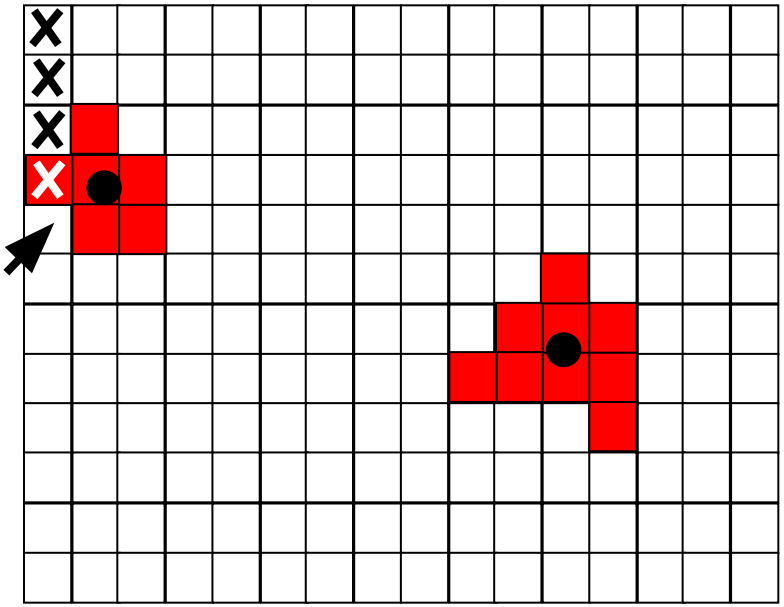
BeadFinder - Depth First Search

- Use boolean[][] array to mark visited
- Traverse image pixel by pixel
  - Dark pixel
    - Mark as visited, continue
  - Light pixel
    - Create new blob, call DFS
- DFS algorithm
  - Base case: simply return if
    - Pixel out-of-bounds
    - Pixel has been visited
    - Pixel is dark (and mark as visited)
  - Add pixel to current blob, mark as visited
  - Recursively visit up, down, left, and right neighbors



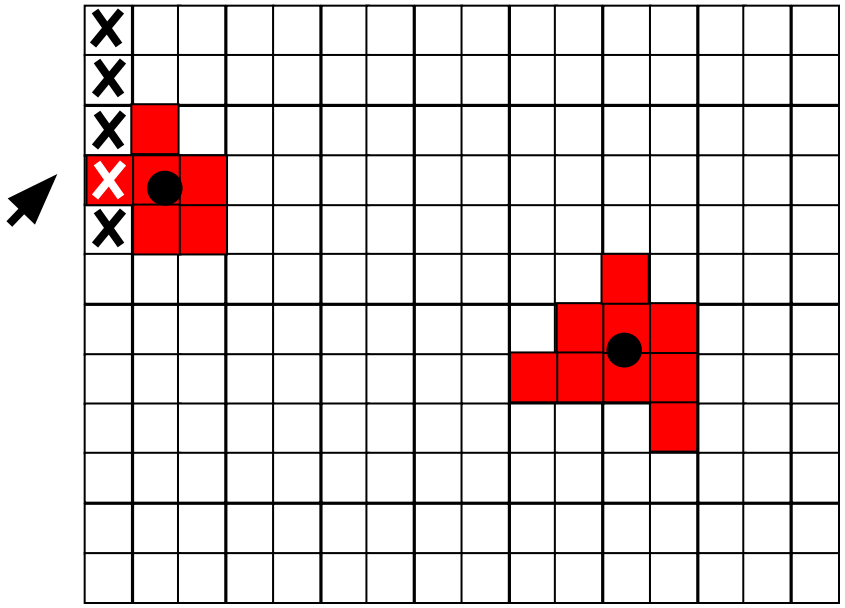
BeadFinder - Depth First Search

- Use boolean[][] array to mark visited
- Traverse image pixel by pixel
  - Dark pixel
    - Mark as visited, continue
  - Light pixel
    - Create new blob, call DFS
- DFS algorithm
  - Base case: simply return if
    - Pixel out-of-bounds
    - Pixel has been visited
    - Pixel is dark (and mark as visited)
  - Add pixel to current blob, mark as visited
  - Recursively visit up, down, left, and right neighbors



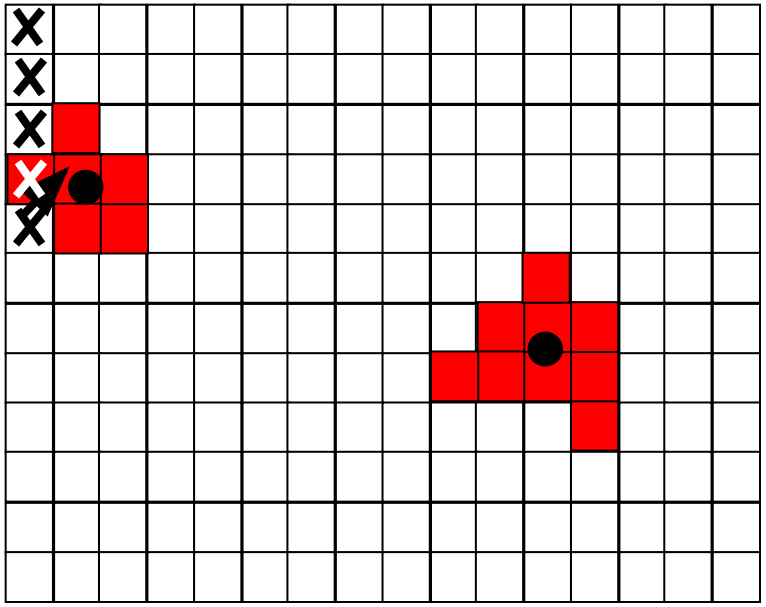
BeadFinder - Depth First Search

- Use boolean[][] array to mark visited
- Traverse image pixel by pixel
  - Dark pixel
    - Mark as visited, continue
  - Light pixel
    - Create new blob, call DFS
- DFS algorithm
  - Base case: simply return if
    - Pixel out-of-bounds
    - Pixel has been visited
    - Pixel is dark (and mark as visited)
  - Add pixel to current blob, mark as visited
  - Recursively visit up, down, left, and right neighbors



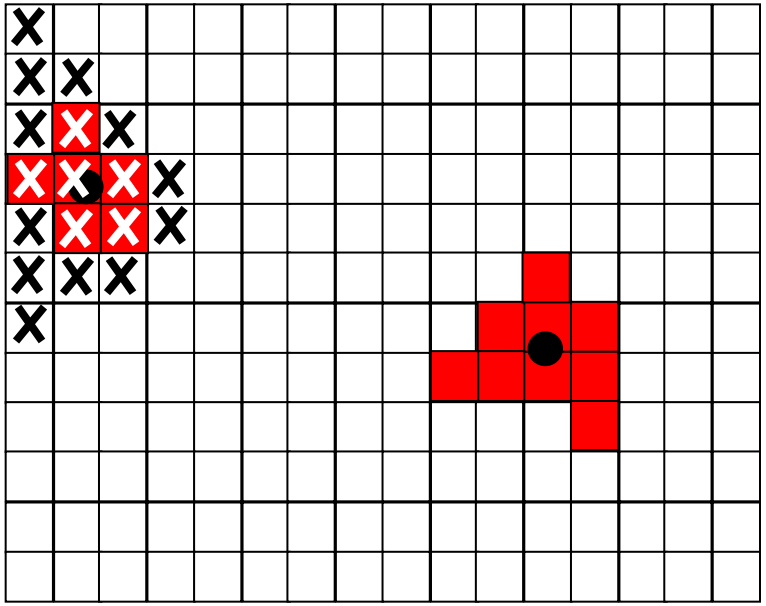
BeadFinder - Depth First Search

- Use boolean[][] array to mark visited
- Traverse image pixel by pixel
  - Dark pixel
    - Mark as visited, continue
  - Light pixel
    - Create new blob, call DFS
- DFS algorithm
  - Base case: simply return if
    - Pixel out-of-bounds
    - Pixel has been visited
    - Pixel is dark (and mark as visited)
  - Add pixel to current blob, mark as visited
  - Recursively visit up, down, left, and right neighbors



BeadFinder - Depth First Search

- Use boolean[][] array to mark visited
- Traverse image pixel by pixel
  - Dark pixel
    - Mark as visited, continue
  - Light pixel
    - Create new blob, call DFS
- DFS algorithm
  - Base case: simply return if
    - Pixel out-of-bounds
    - Pixel has been visited
    - Pixel is dark (and mark as visited)
  - Add pixel to current blob, mark as visited
  - Recursively visit up, down, left, and right neighbors



BeadFinder Challenges

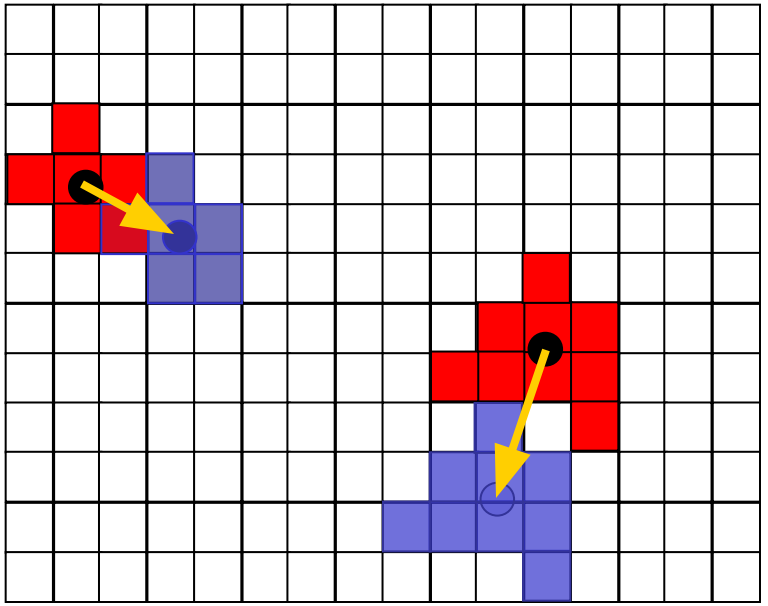
- Data structure for the collection of blobs
  - Store them any way you like
  - But be aware of memory use and timing

BeadFinder Challenges

- Data structure for the collection of blobs
  - Store them any way you like
  - But be aware of memory use and timing
- Array of blobs?
  - But how big should the array be?
- Linked list of blobs?
  - Memory efficient, but harder to implement
  - Avoid traversing whole list to add a blob!
- Anything else?
  - Submit your (extra) object classes

BeadTracker.java

- Track beads between successive images
- Single main function
  - Take in a series of images
  - Output distance traversed by all beads for each time-step
    - For each bead found at time t+1, find closest bead at time t and calculate distance
      - Not the other way around!
      - Don’t include if distance > 25 pixels (new bead)



BeadTracker Challenges

- Reading multiple input files
  - `java-introcs BeadTracker 25 180.0 25.0 run_1/*.jpg`
  - Expands files in alphabetical order
  - End up as `args[0], args[1], …`
- Avoiding running out of memory
  - How?
- Recompiling
  - Recompile if Blob or BeadFinder change

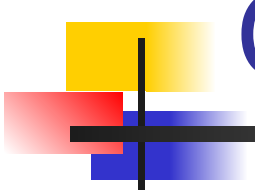
BeadTracker Challenges

- Reading multiple input files
  - `java-introcs BeadTracker 25 180.0 25.0 run_1/*.jpg`
  - Expands files in alphabetical order
  - End up as `args[0], args[1], …`
- Avoiding running out of memory
  - Do *not* open all picture files at same time
  - Only two need to be open at a time
- Recompiling
  - Recompile if Blob or BeadFinder change

Avogadro.java

- Analyze Brownian motion of all calculated displacements
  - Lots of crazy formulas, all given, pretty straightforward
  - Be careful about units in the math, convert pixels to meters, etc.
- Can test without the other parts working
  - We provide sample input files
  - Can work on it while waiting for help

Conclusion: Final Tips



- Avoiding subtle bugs in BeadFinder
  - Double check what happens at corner cases (e.g. at boundary pixels, or when luminance == tau, or mass == cutoff)
- Common errors in BeadFinder
  - `NullPointerException`
  - `StackOverflowError` (e.g., if no base case)
  - No output (need to add print statements)
- Look at *Possible Progress Steps*
  - Click ▸ to expand!

Conclusion: Final Tips

- Avoid *magic numbers*
  - Define constants
- No Checkstyle or other errors/warnings
- Testing with a `main()`
- There is a limit of **twenty (20) times** that you may click the Check Submitted Files to receive feedback from the TigerFile auto-grader
  - So, test locally! I.e., on your laptop before using TigerFile to run test cases

Conclusion: Final Tips

- Timing analysis - *doubling method!*
  - Wild cards
    - The frames use the following naming convention:
      - `frame00000.jpg, frame00001.jpg …`
        `frame00198.jpg, frame00199.jpg`
    - On command line:
      - 10 frames? `run_1/frame0000*.jpg`
      - 20 frames? `run_1/frame000[01]*.jpg`
      - 40 frames? `run_1/frame000[0123]*.jpg`
      - 100 frames? `run_1/frame000*.jpg`
      - 200 frames? `run_1/frame*.jpg`
      - 400 frames? `run_1/frame*.jpg run_1/frame*.jpg`

Thank you

References

- Prof. Ibrahim Albluwi
- https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/Robert_Brown_(botanist).jpg/220px-Robert_Brown_(botanist).jpg
- https://cdn.miniphysics.com/wp-content/uploads/2011/01/brownianmotion.gif
- https://upload.wikimedia.org/wikipedia/commons/d/d3/Albert_Einstein_Head.jpg
- https://en.wikipedia.org/wiki/Jean_Baptiste_Perrin#/media/File:Jean_Perrin_1926.jpg