CS 376A
Digital Image Processing

02 / 08 / 2023

Instructor: Michael Eckmann

Today's Topics

- Questions / Comments?
- Morphological processing
  - Review code for dilate and erode
  - Implement Open and Close
- Neighborhood definitions
  - 4 adjacency
  - 8 adjacency
- Connected Component labelling problem
  - Solution using a union-find data structure and a two pass algorithm through the image

binary morphology

- I wrote code to
  - read ppm files into an RGBImage
  - write ppm files from an RGBImage
- For binary morphology I wrote code to
  - dilate
  - erode
- an RGBImage (that is assumed to only have 0,0,0 and 255,255,255 pixels) by a 2d structuring element (of booleans)
- Let's try a few structuring elements and images
- Also, let's write open and close and try those.

Neighborhoods

- 4-adjacency
- 8-adjacency

Connected component problem

- Problem:
  - Give a label to each connected component of an image (consider black/white images only for now where black is background, white is foreground). A connected component is a set of connected foreground pixels.

Connected components

- The following connected component labelling algorithm uses 4 adjacency.
- It uses a union-find data structure which has these operations
  - Union operation takes two labels and the union-find data structure and joins the two sets that have each of those labels into one set.
  - Find operation takes one label and the union-find data structure and finds the set containing that label and returns the smallest number label in that set.

Connected components

- One way this can be accomplished is by implementing this union-find data structure as an array that represents a forest of trees.
- Each tree in the forest represents an equivalence class (the labels in the nodes form a set that may be different labels for the SAME connected component.)
- The indices of the array are the labels. An element in the array is the parent label of the index label.

- Example on the board of what is stored in an array for a forest of trees.
- And pseudocode for union and find.

Connected components

- Also need an operation to just add a label as a tree of one node.
- Pseudocode for union and find.
  - Union
  - parameters: labelOne, labelTwo, parentArray
    - loop until find root of labelOne
    - loop until find root of labelTwo
    - If those two roots are not the same, make one the parent of the other.
  - Find
  - parameters: label, parentArray
    - loop until find root of label
    - return root

Connected components

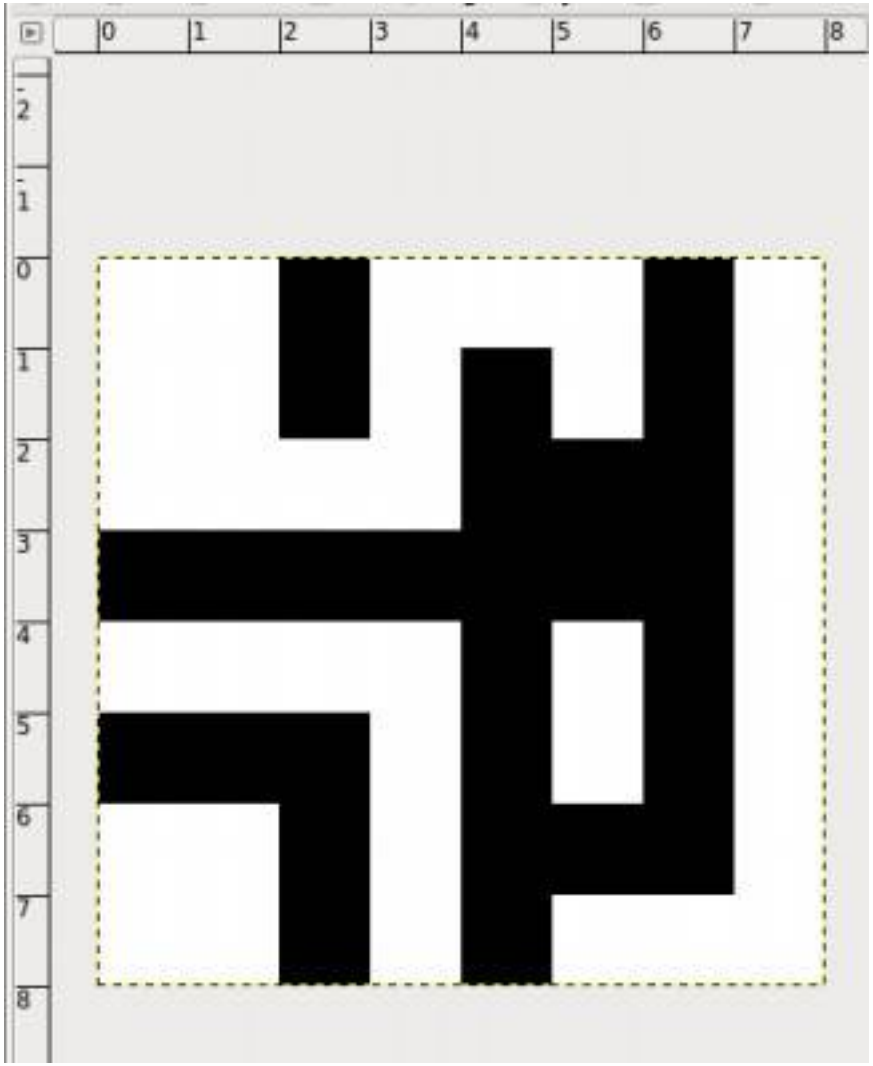
- PASS 1: travel each row from left to right
  - if see a foreground pixel, examine label of pixel above and to left
  - if only one is labeled, then label the current pixel that label
  - if both are labeled, then label the current pixel the lesser of the two labels
  - if neither above nor left pixel has labels (i.e. they are background pixels), then pick a new label for current pixel
  - if both above and left pixel have different labels, enter the equivalence class into the union-find data structure (perform a union of the two sets)
- After PASS 1, all foreground pixels have been labeled and we have a set of equivalence classes for all labels

Connected components

- PASS 2:
  - relabel the labels of the pixels with the root label of the equivalence class in the union-find data structure (using the find algorithm to determine this root label)

- Example run of this algorithm using image on next slide.

Connected components



Connected components

- Consider extending this to color or grayscale images.

- The part that needs to change in the algorithm is the assumption that all pixels are either background or foreground and that all foreground pixels are the same.

Connected Components

- Flood fill algorithms
  - select an unlabeled foreground pixel – give it a new label
    - label each of its neighbors' foreground pixels with same label
    - continue until no neighbors have unlabeled foreground pixels
  - repeat steps above until no unlabeled foreground pixels remain

Regions and their properties

- We may cover these later in the semester --- these do not produce new images, but instead describe the content of images
- Once regions are labeled independently, further processing can take place such as
    - describing each region by
        - some contour representation
        - determining and describing the shape via a skeleton
        - describing it quantitatively
            - area
            - centroid
            - shape and orientation of the axes
            - etc.

Border extraction

- Extracting border of region(s)
- Define border pixels to be those that are adjacent to pixels in a region but are not part of the region.

- Example of an image and its border image

- Consider one or a combination of two or more of the following operations:
  - erode (fits), dilate (hits),
  - open (erode then dilate), close (dilate then erode)
  - and, or, not, minus (1-0 = 1, 1-1=0-0=0-1=0)

Border extraction

- Let's implement your ideas.