

CS 331

Computer Vision

09 / 22 / 2026

Instructor: Michael Eckmann

Today's Topics

- Questions? / Comments?
- Binary Image Morphology operations and uses
- Connected Components algorithm
- Some examples of opening, closing and border using Binary Image Morphology
- Creating a border, using binary image operations

Morphology

- Remind me what you know about Binary Image Morphology

structuring element

- a structuring element is typically a small binary image representing some shape. A structuring element has an origin.
 - e.g. a 3x3 square of all 1-pixels, with center as origin
 - a 5x5 binary image with a diamond shape of 1-pixels, others 0's
- a structuring element is applied to a binary image by hovering the origin pixel over each pixel in the image one at a time
 - only the 1-pixels in the structuring element matter (the 0-pixels of the structuring element are not compared to anything)
 - the structuring element **fits** means that *all* the 1-pixels in the structuring element correspond to 1-pixels in the image
 - the structuring element **hits** means that *at least one* 1-pixel in the structuring element corresponds to a 1-pixel in the image

binary morphology

- binary image morphological operations
 - image is f , structuring element is s , resulting image is g
 - **dilation** – increases the size of regions
 - f dilated by s results in g where g contains a 1 where s hits f , 0 otherwise
 - **erosion** – decreases the size of regions
 - f eroded by s results in g where g contains a 1 where s fits f , 0 otherwise

binary morphology

- binary image morphological operations
 - What's the general affect of dilation?
 - What's the general affect of erosion?

binary morphology

- binary image morphological operations
 - image is f , structuring element is s , resulting image is g
 - **closing** – closes up holes within regions
 - f closed by s results in g , which is the result of f dilated by s and then eroded by s
 - Dilate then Erode
 - **opening** – get rid of jutting out portions of regions
 - f opened by s results in g , which is the result of f eroded by s and then dilated by s
 - Erode then Dilate
- Both opening and closing are idempotent
 - this means after one closing of f by s , further closings by s do not change the result
 - same for opening

binary morphology

- binary image morphological operations
 - The result of closing would be to keep the region about the same size while filling in gaps
 - The result of opening would be to keep the region about the same size while removing jutting out pixels

binary morphology

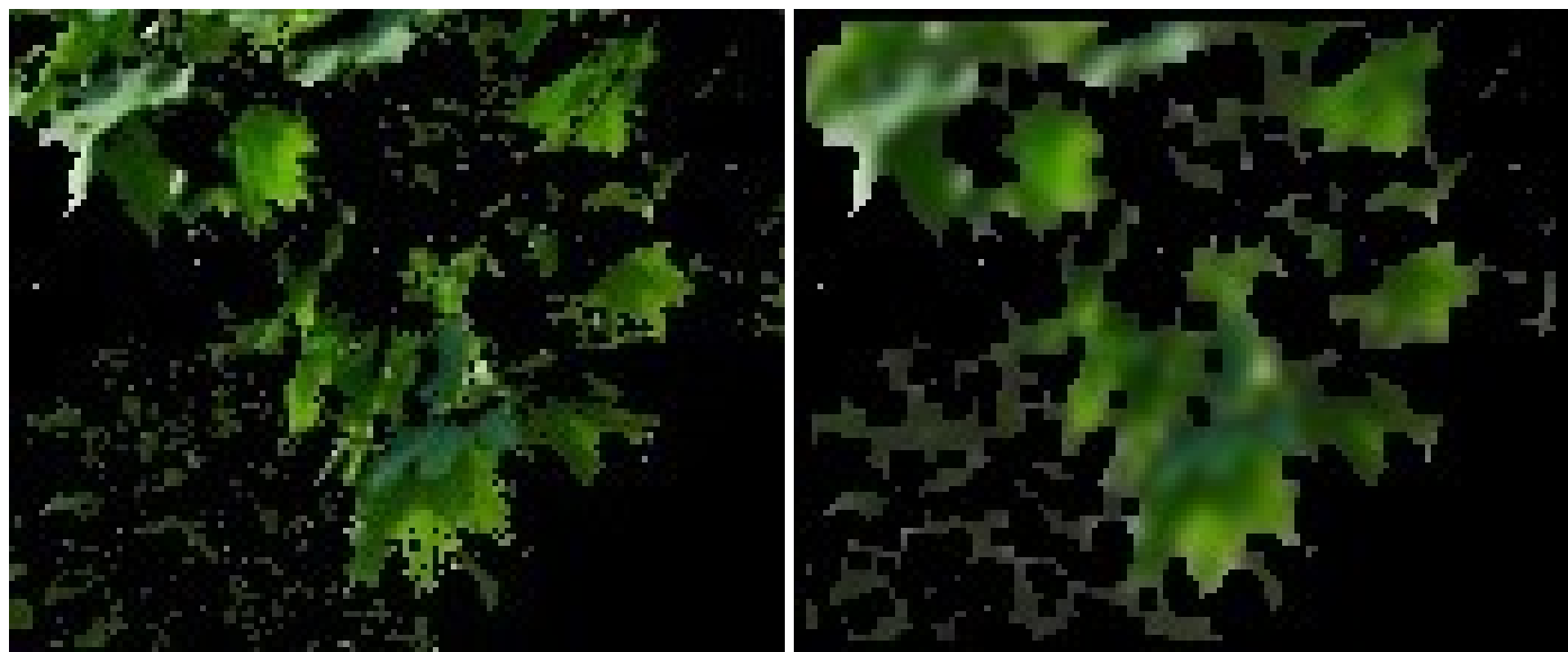
- binary image morphological operations
 - Demo program for dilate, erode, open, close
 - First let me show something with jugged out pixels and result of open on it
 - Second let me show something with holes and result of close on it

binary morphology

- binary image morphological use:
 - For that green image let's close it with a 3x3 structuring element of all 1's --- I tweaked the idea of binary morphology for this --- if an output pixel was to be written as “on” or “white”, I made it the average of the 8 “on” neighbors. That is I averaged the nearby greens and itself.

binary morphology

- Result (on the right) of closing the left image with a 3x3 all 1's structuring element

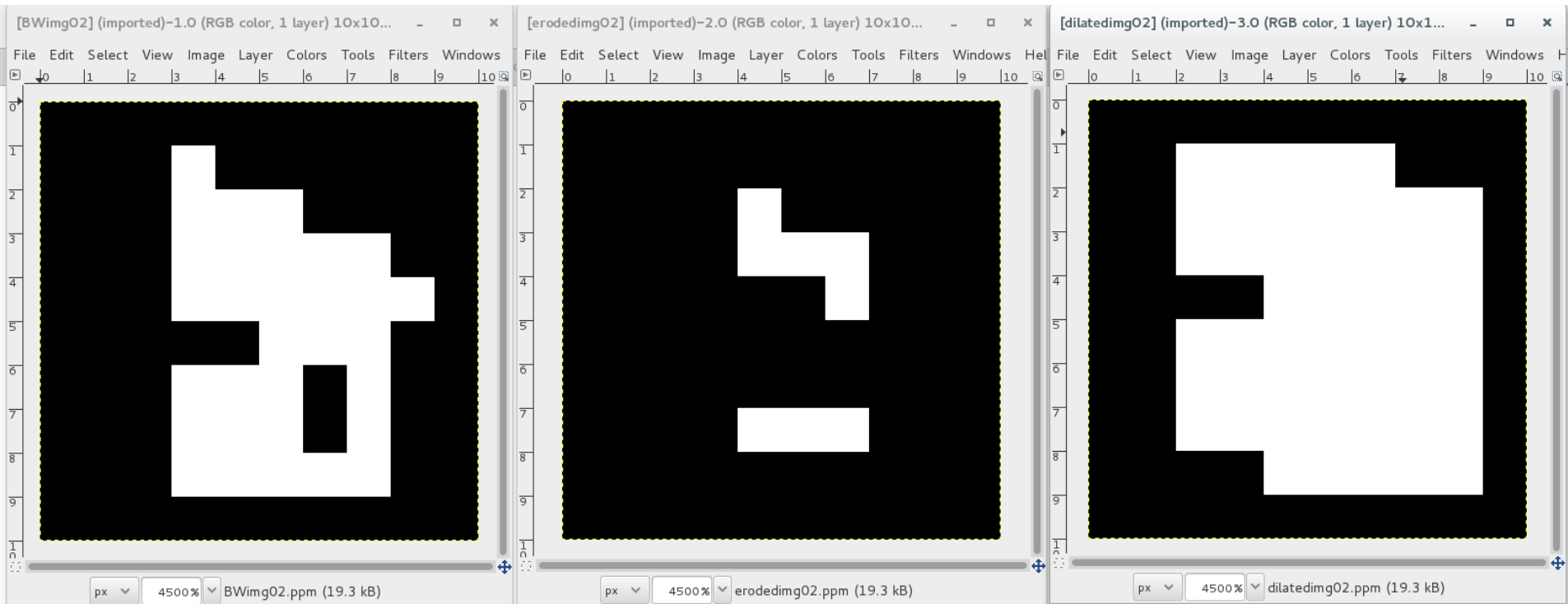


binary morphology

- Other simple operations that can be done to binary images
 - f and g are binary images of the same dimensions
 - AND: $f \text{ AND } g$ results in h where h contains a 1 iff f and g both contain a 1
 - OR: $f \text{ OR } g$ results in h where h contains a 1 if either f and/or g contain a 1
 - NOT: $\text{NOT } f$ results in h where all 0's in f result in 1's in h and all 1's in f result in 0's in h
 - MINUS: $f \text{ MINUS } g$ results in h where
 - $1 \text{ minus } 0 = 1$
 - all others 0

Connected components

- Once we have cleaned up our image using morphology, we then might like to figure out which pixels belong to which regions.
- A pixel belonging to a region means it is connected to other pixels in the same region.
- Left and right images below contain only 1 region, middle image contains 2.



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

- We can consider either neighbors as 4-adjacent or 8-adjacent
 - 4-adjacent means a pixel is connected if it is adjacent to another pixel in the region either up, down, left or right
 - 8-adjacent means a pixel is connected if it is adjacent to another pixel in the region either up, down, left, right or diagonally up/left, up/right, down/left, down/right

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

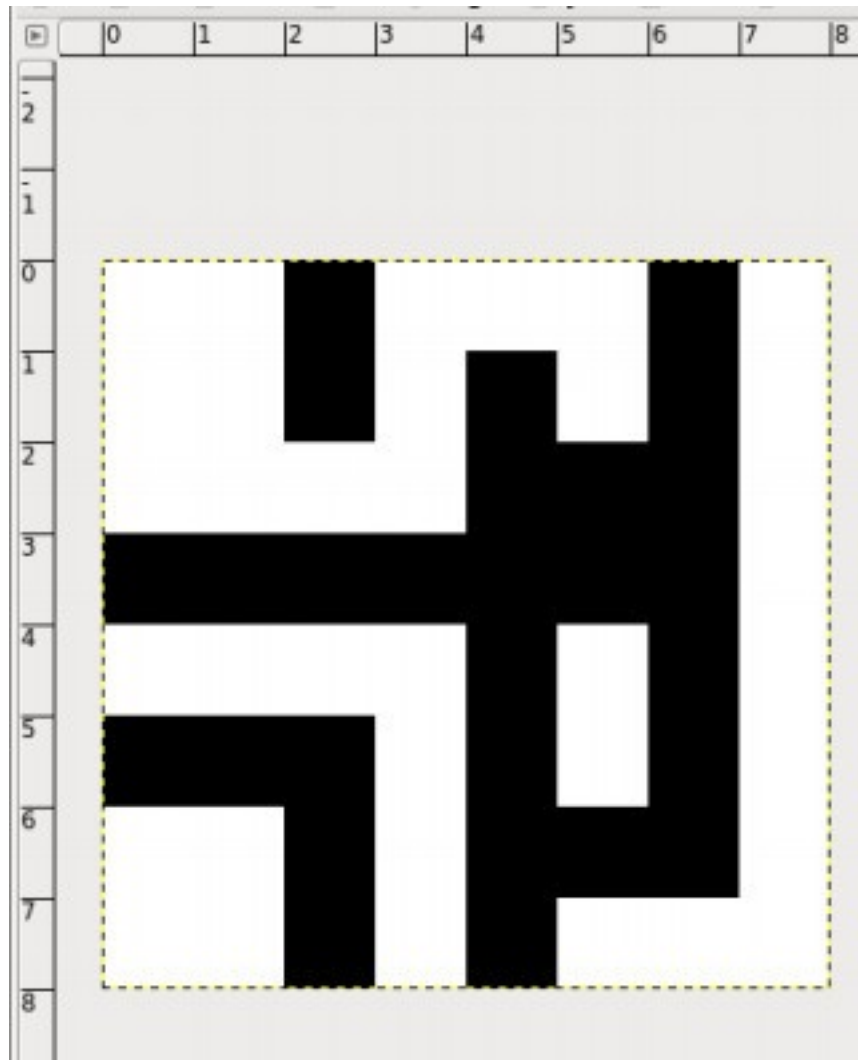
Connected components

- 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 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



Michael Eckmann - Skidmore
College - CS 331 - Fall 2026

Binary Morphology program

- Let me demonstrate binary morphology on a binary image of Abraham Lincoln.
- I have 3 versions of this image – one that is ideal, one that has juttied out pixels and one that is noisy.
- I have implemented dilate, erode, open, close and others
- Let's open the one that has juttied out pixels (with a 10x10 structuring element)
- Let's close the one that is noisy (with a 10x10 structuring element)

Binary Morphology program

- Recall that we know how to do dilate, erode, open, close, and, or, not, and minus operations. What combination of those do you think might produce a border?

Binary Morphology program

- Recall that we know how to do dilate, erode, open, close, and, or, not, and minus operations. What combination of those do you think might produce a border?
- I implemented it as erode followed by minus
- I also implemented it as dilate followed by minus