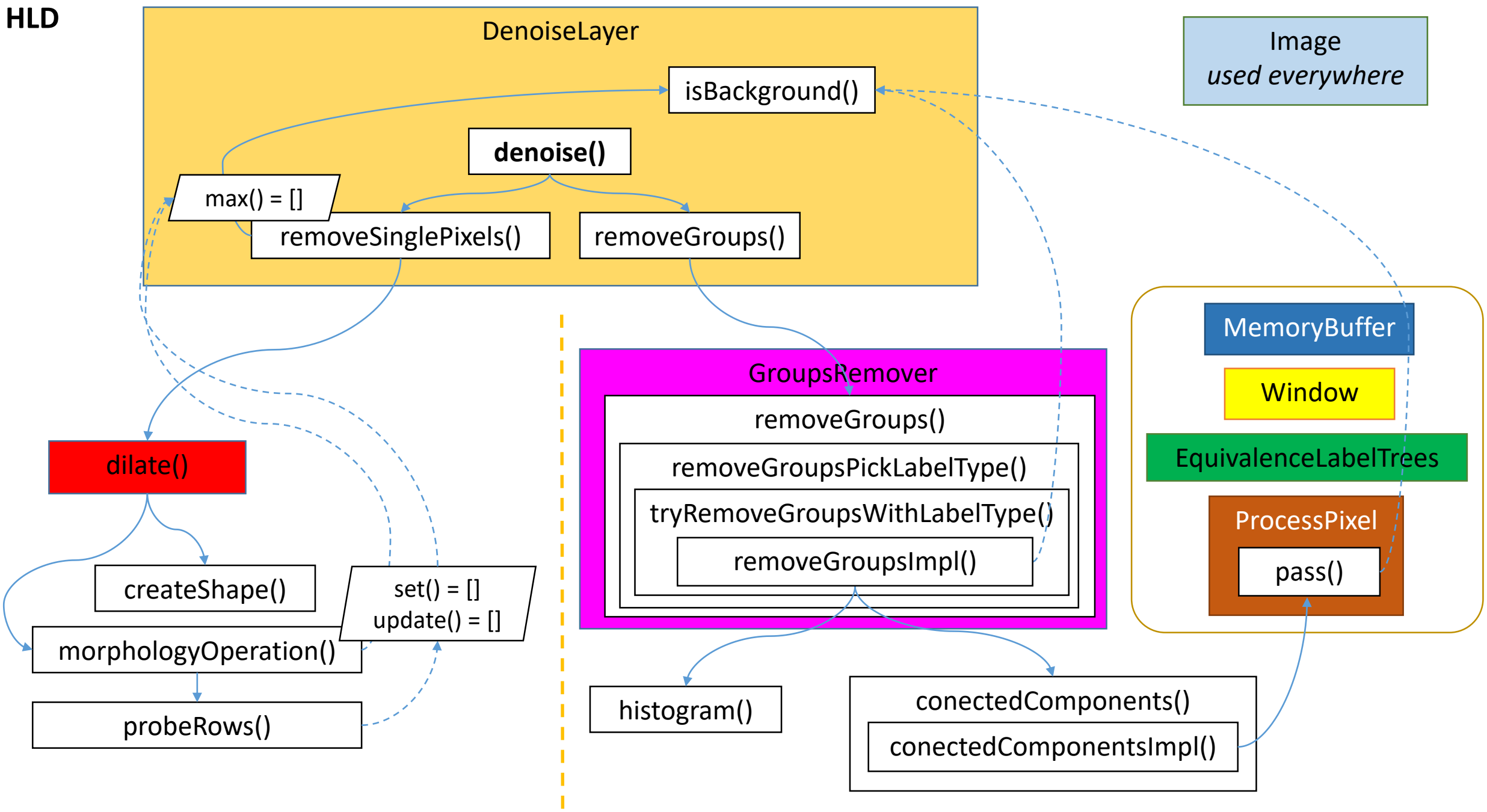
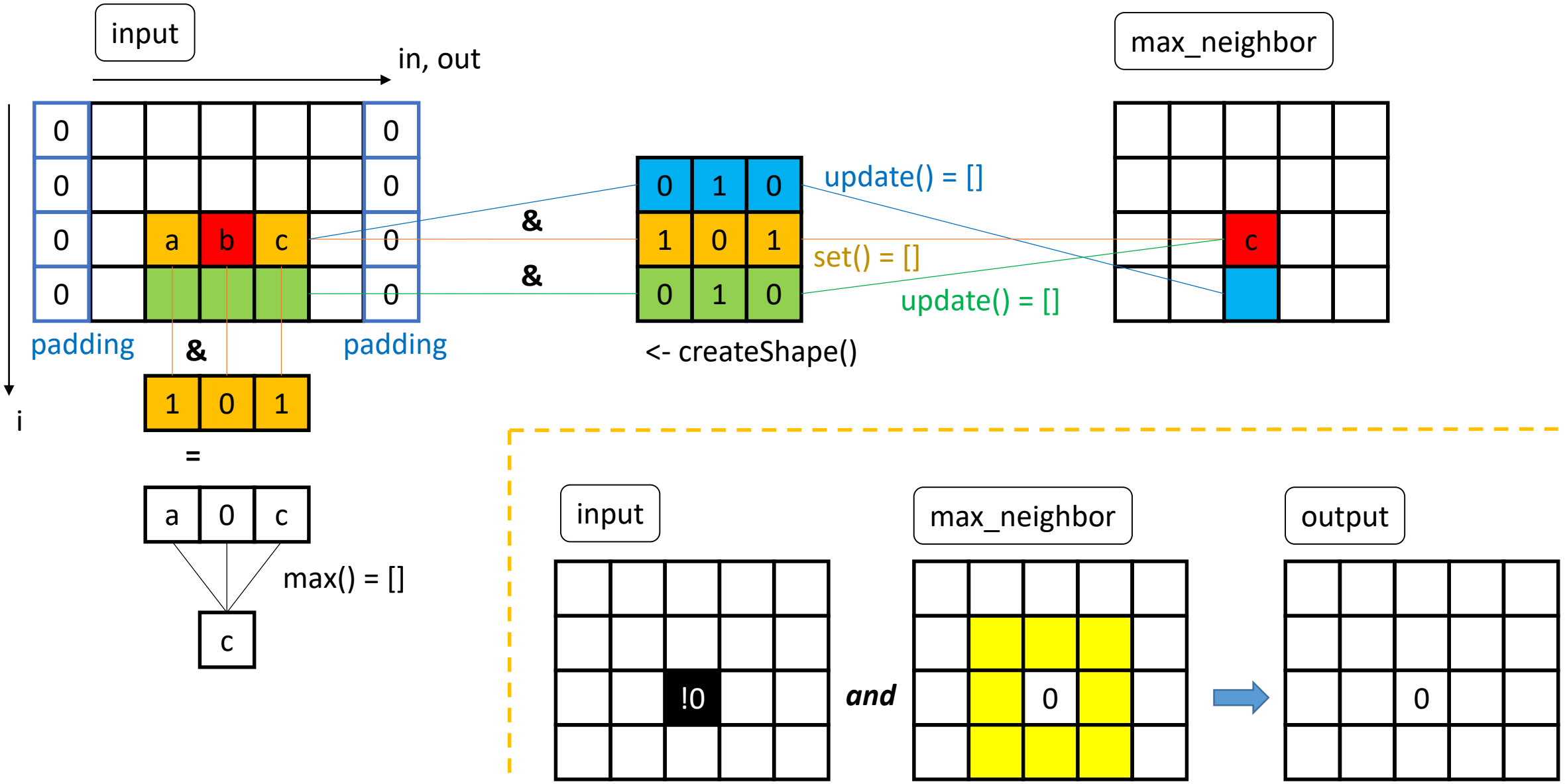
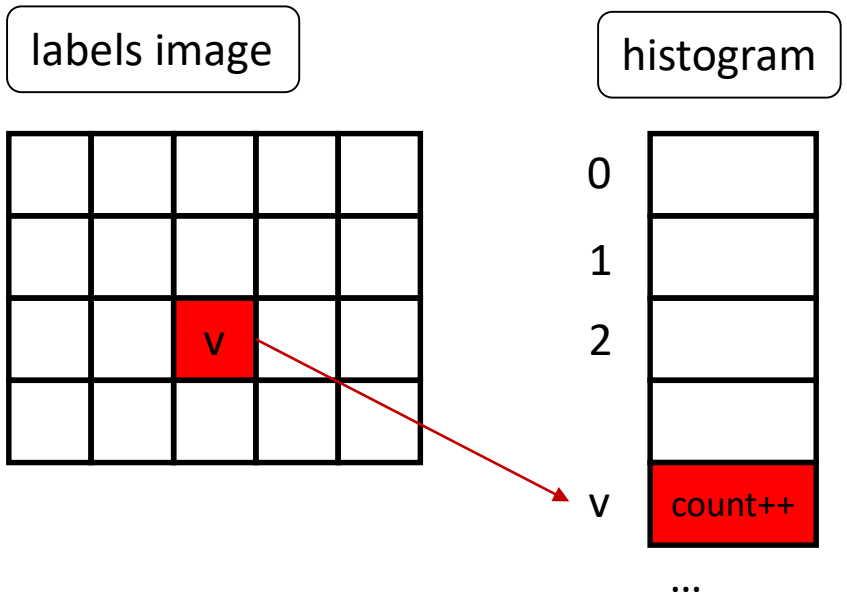
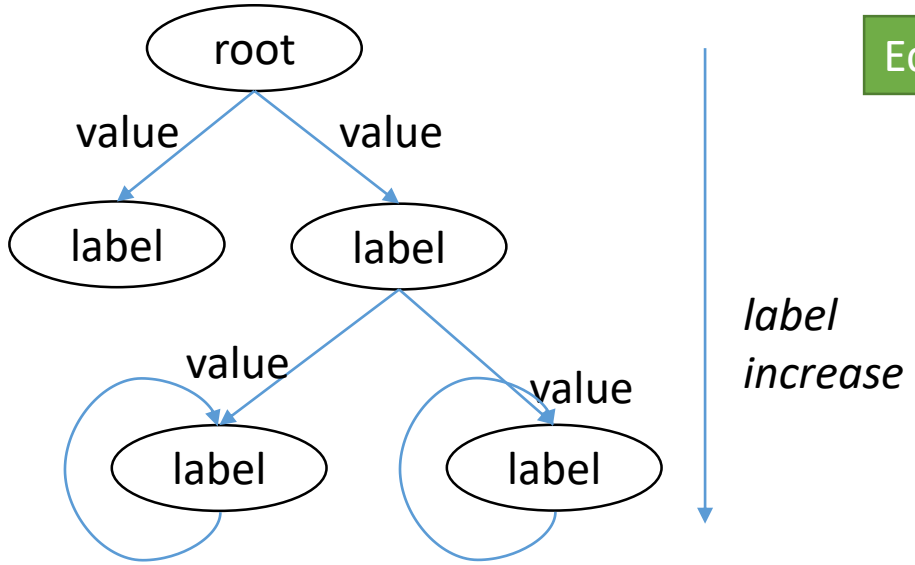

HLD
DenoiseLayer
isBackground()
denoise()
max() = []
removeSinglePixels()
removeGroups()
Image
used everywhere
MemoryBuffer
Window
EquivalenceLabelTrees
ProcessPixel
pass()
GroupsRemover
removeGroups()
removeGroupsPickLabelType()
tryRemoveGroupsWithLabelType()
removeGroupsImpl()
dilate()
createShape()
set() = []
update() = []
morphologyOperation()
probeRows()
histogram()
conectedComponents()
conectedComponentsImpl()

removeSinglePixels() algorithm



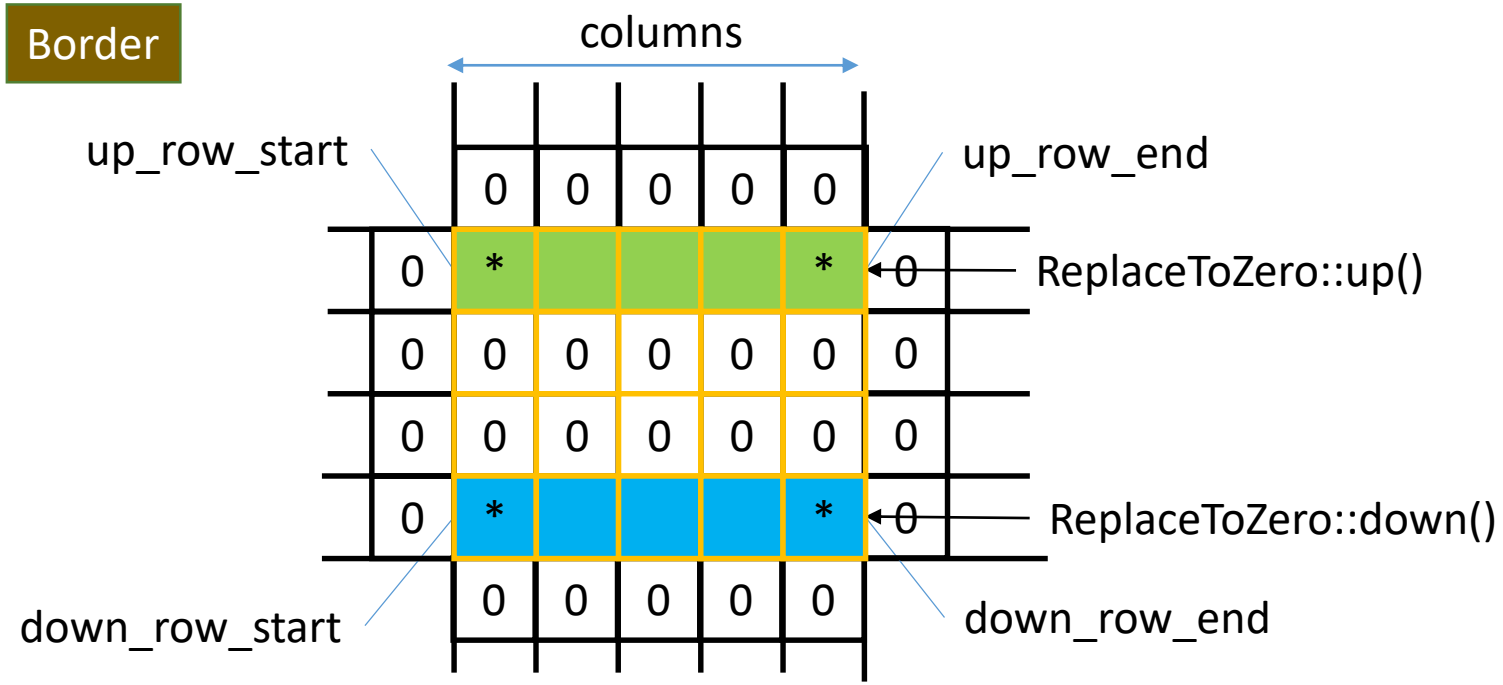
Types used in removeGroups() algorithm



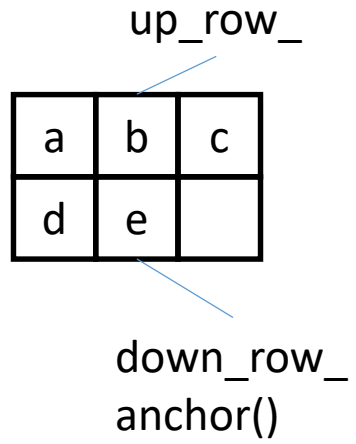


EquivalenceLabelTrees

Border



Window



removeGroups() algorithm

- **`GroupsRemover::removeGroups()`** <- wrapper for `removeGroupsPickLabelType()`, called either with ConnectivityType::Way4 or ConnectivityType::Way8
- **`removeGroupsPickLabelType()`** <- tries to allocate `EquivalanceLabelTrees<uint16_t>`, otherwise - `EquivalanceLabelTrees<uint32_t>`, and then calls `tryRemoveGroupsWithLabelType()` with pre-allocated label trees
- **`tryRemoveGroupsWithLabelType()`** <- try-catch wrapper for `removeGroupsImpl()`
- **`removeGroupsImpl()`**
  - obtains `labels, group_count = connectedComponents()`
  - calculates `group_sizes = histogram(lables)`
  - gets `noise_lables_table` as `group_sizes` with a value less than MGS
  - converts input image, removing all cells belonging to `noise_lables_table`

- **`connectedComponents()`**
  - allocates `labels` image (same size as image)
  - calls `connectedComponentsImpl()` and returns std::pair from it
- **`connectedComponentsImpl()`**
  - scans row 0
  - iterates rows 1...row
    - Iterates columns 1…col-2; scan col-1
    - On each iteration applies `ProcessPixel::pass()` with sliding `Window` and get `label_trees`
  - converts `label_trees` to `lables_map`
  - makes `lables` image as `lables[iter] = lables_map[image[iter]]`

removeGroups() algorithm