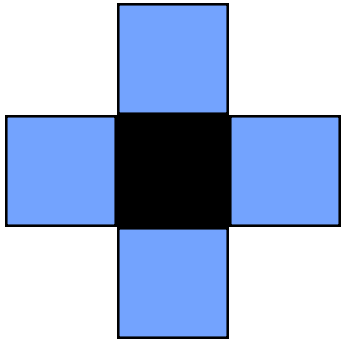
Connected Components

- Basic definitions
  - Connectivity, Adjacency, Connected Components
  - Background/Foreground, Boundaries
  - Run-length encoding
- Component Labeling
  - Recursive algorithm
  - Two-scan algorithm
- Chain Codes
- Integral Images
- Histograms
  - Gray and Color Histograms
  - Edge Histograms

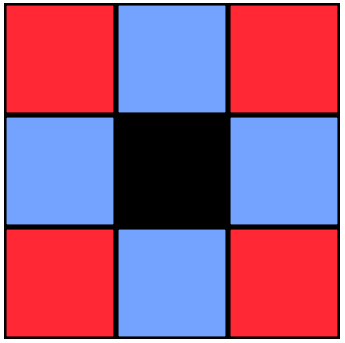
Zoran Duric

4(8) Connectivity

- Definition: Given a pixel (i,j) its 4-neighbors are the points (i',j') such that $|i-i'| + |j-j'| = 1$
  - the 4-neighbors are (i±i, j) and (i,j±1)

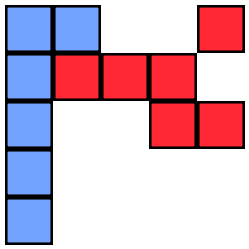


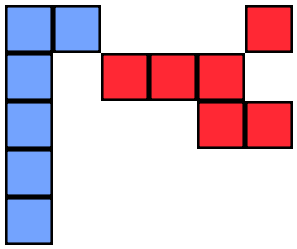
- Definition: Given a pixel (i,j) its 8-neighbors are the points (i',j') such that $\max(|i-i'|,|j-j'|) = 1$
  - the 8- neighbors are (i, j±1), (i±1, j) and (i±1, j±1)



Adjacency

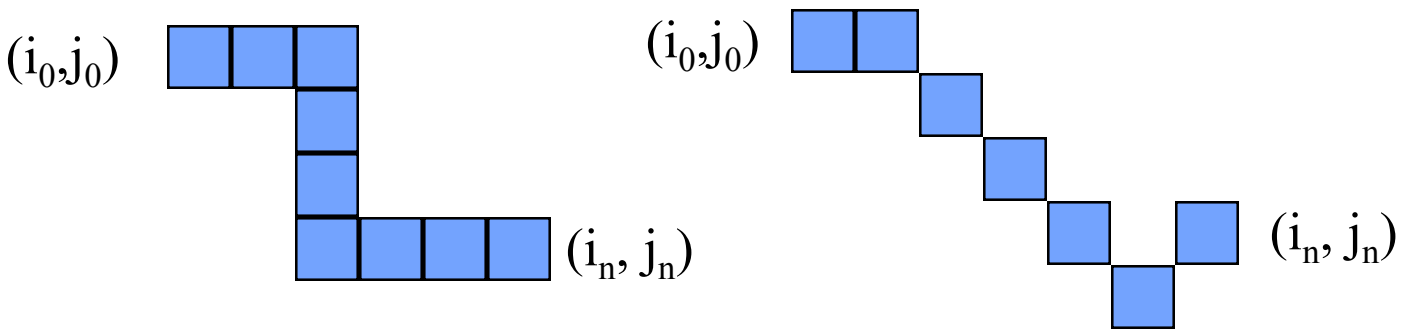
- Definition: Given two disjoint sets of pixels, A and B, A is 4-(8) adjacent to B is there is a pixel in A that is a 4-(8) neighbor of a pixel in B





Connected components

- Definition: A 4-(8)path from pixel (i_0,j_0) to (i_n,j_n) is a sequence of pixels (i_0,j_0) (i_1,j_1) (i_2,j_2) , ... (i_n,j_n) such that (i_k, j_k) is a 4-(8) neighbor of (i_{k+1}, j_{k+1}), for k = 0, ..., n-1



Every 4-path is an 8-path!

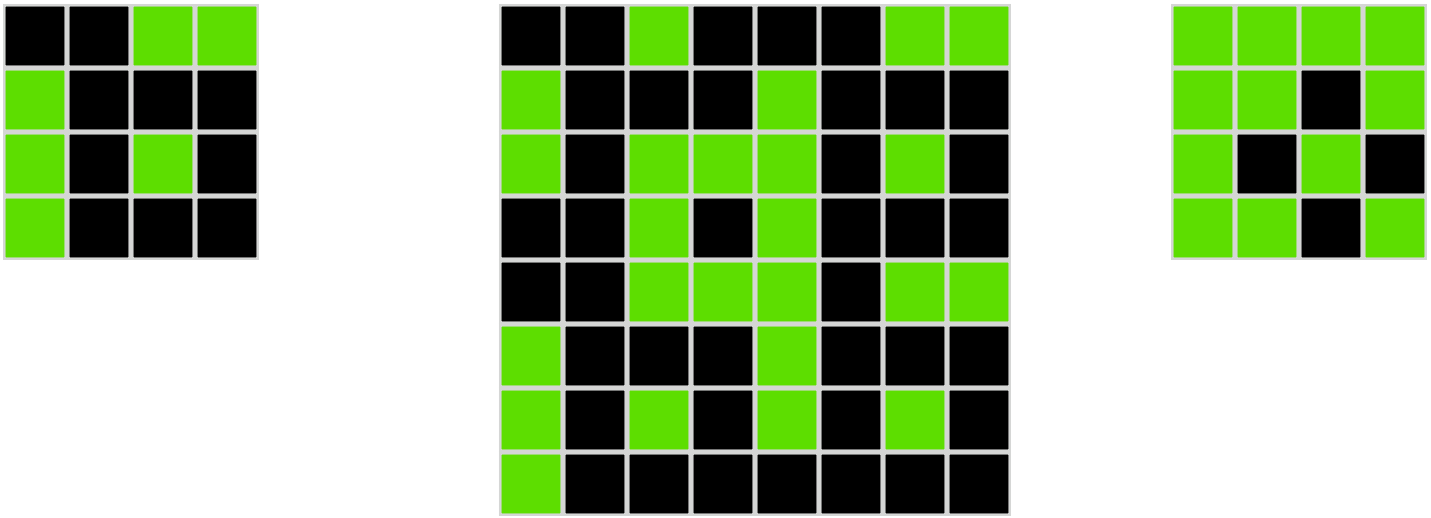
Connected components

- Definition: Given a binary image, B, the set of all 1's is called the foreground and is denoted by S
- Definition: Given a pixel p in S, p is 4-(8) connected to q in S if there is a path from p to q consisting only of points from S.
- The relation "is-connected-to" is an equivalence relation
  - Reflexive - p is connected to itself by a path of length 0
  - Symmetric - if p is connected to q, then q is connected to p by the reverse path
  - Transitive - if p is connected to q and q is connected to r, then p is connected to r by concatenation of the paths from p to q and q to r

Connected components

- Since the "is-connected-to" relation is an equivalence relation, it partitions the set S into a set of equivalence classes or components
  - these are called connected components
- Definition: $\bar{S}$ is the complement of S - it is the set of all pixels in B whose value is 0
  - $\bar{S}$ can also be partitioned into a set of connected components
  - Regard the image as being surrounded by a frame of 0's
  - The component(s) of $\bar{S}$ that are adjacent to this frame is called the background of B.
  - All other components of $\bar{S}$ are called holes

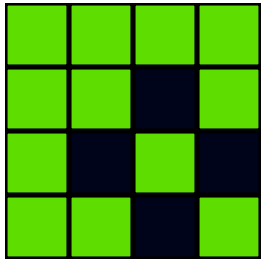
Examples - Black = 1, Green = 0



How many 4- (8) components of S?
What is the background?
Which are the 4- (8) holes?

Background and foreground connectivity

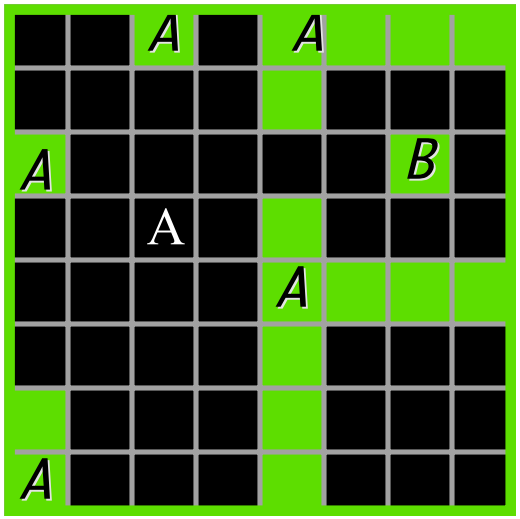
- Use opposite connectivity for the foreground and the background
  - 4-foreground, 8-background: 4 single pixel objects and no holes
  - 4-background, 8-foreground: one 4 pixel object containing a 1 pixel hole

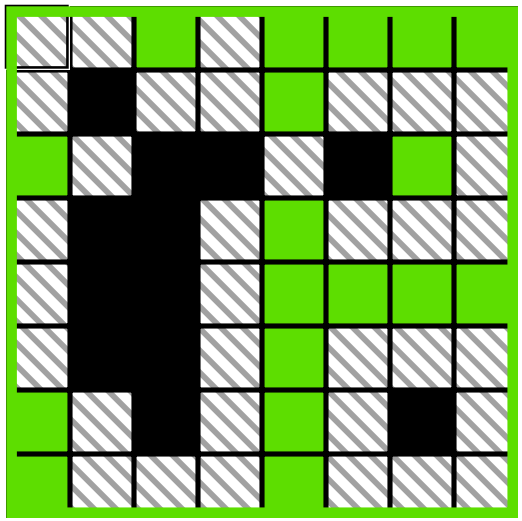


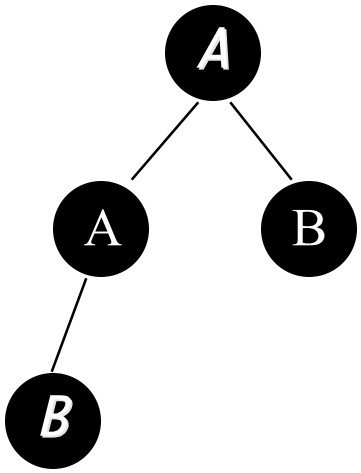
Boundaries

- The ***boundary*** of S is the set of all pixels of S that have 4-neighbors in *S*. The boundary set is denoted as S'.
- The ***interior*** is the set of pixels of S that are not in its boundary: S-S'
- Definition: Region T ***surrounds*** region R (or R is ***inside*** T) if any 4-path from any point of R to the background intersects T
- Theorem: If R and T are two adjacent components, then either R surrounds T or T surrounds R.

Examples





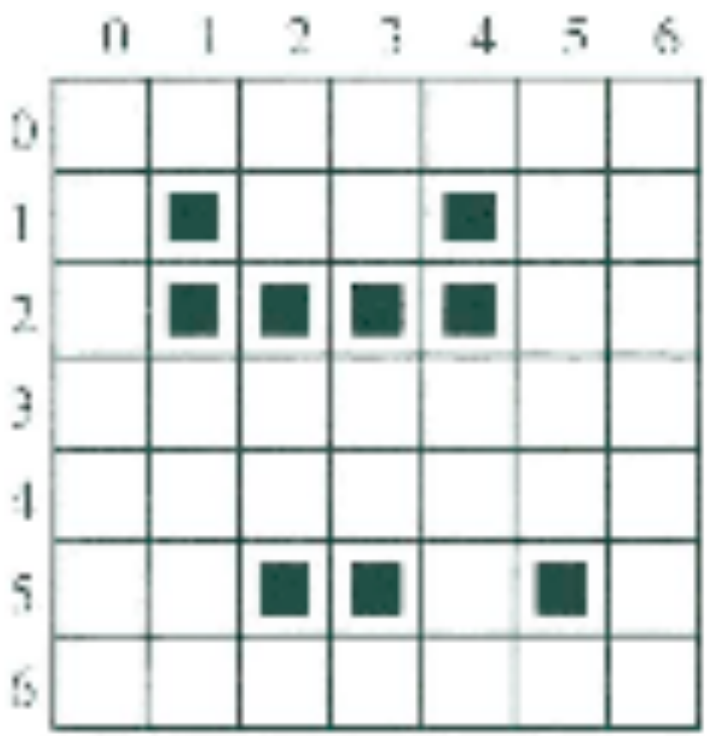


Even levels are components of 0's
The background is at level 0
Odd levels are components of 1's

Run-length Encoding

Used mainly to represent binary images such as faxes.

Various approaches exist, the book represents only foreground.



Foreground is a list of lists.

Each non-zero row is represented by a list.

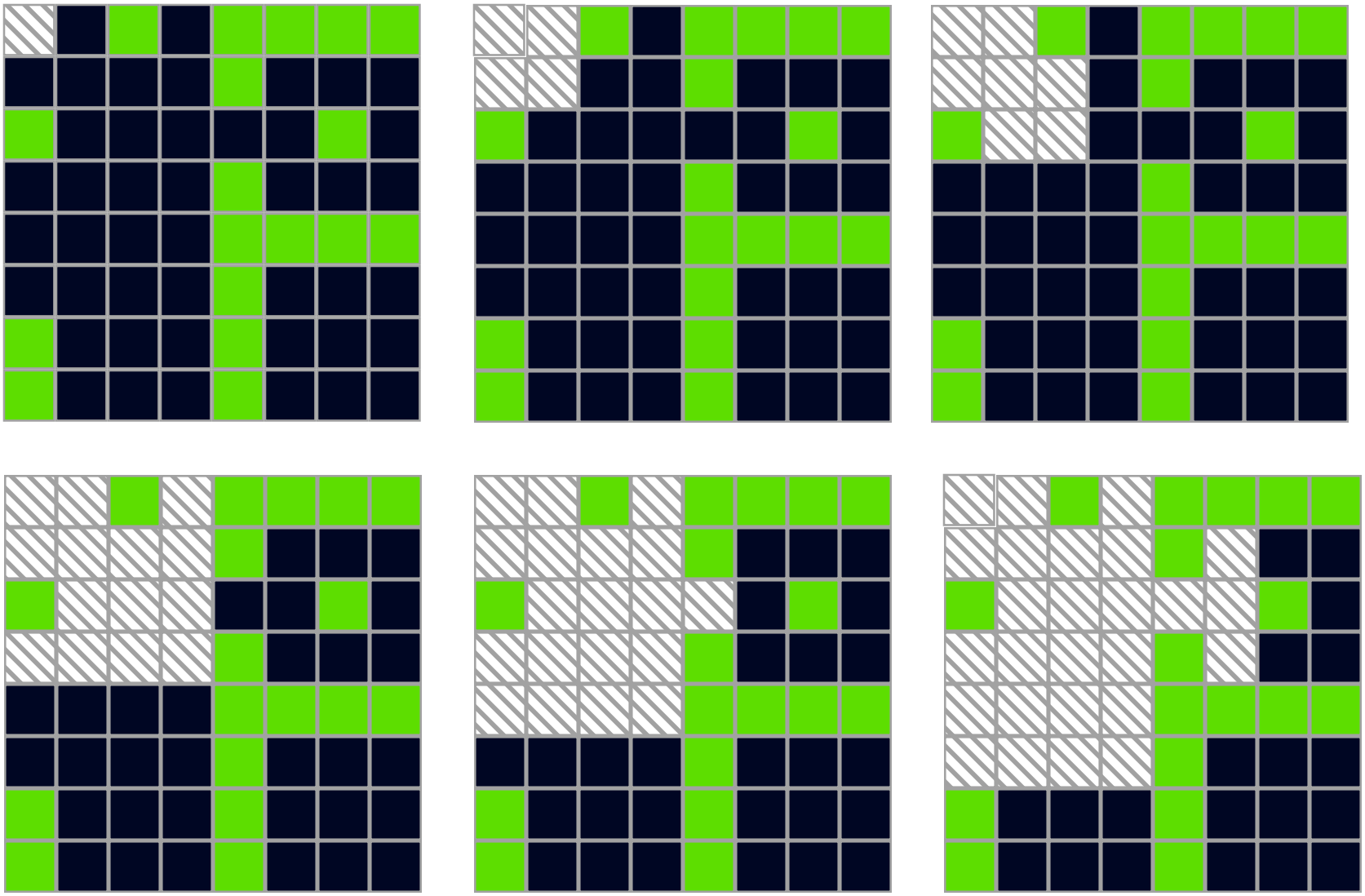
$(r\ b_1e_1\ b_2e_2 \ldots)$ where r is the row and b_ie_i are the beginning and ending column indices for a foreground run

((11144)(214)(52355))

Component labeling

- Given: Binary image B
- Produce: An image in which all of the pixels in each connected component are given a unique label.
- Solution 1: Recursive, depth first labeling
  - Scan the binary image from top to bottom, left to right until encountering a 1 (0).
  - Change that pixel to the next unused component label
  - Recursively visit all (8,4) neighbors of this pixel that are 1's (0's) and mark them with the new label

Example



Recursive Algorithm

1. Create stack S, initially empty
2. Scan the binary image from top to bottom, left to right until encountering a 1 (0).
3. Change that pixel's label to the next unused component label
4. Push the pixel on S (push the coordinates)
5. While S is not empty

    Pop a pixel p from S

    For each unlabeled neighbor of p if it's value is 1(0)

        label it with the current label

        push it on S

Topology Challenge

- How to determine which components of 0's are holes in which components of 1's
- Scan labeled image:
  - When a new label is encountered make it the child of the label on the left

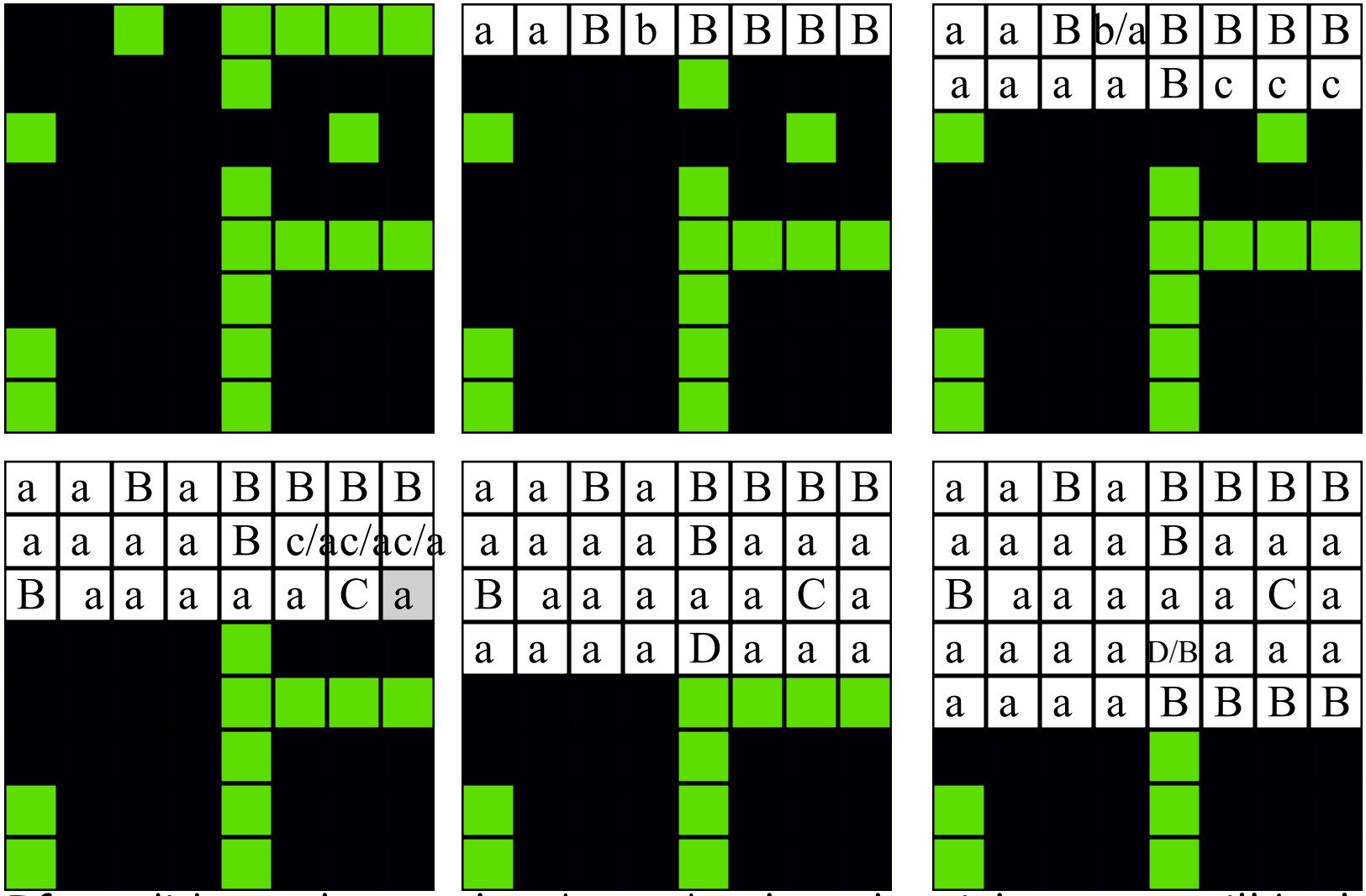
Solution 2 - row scanning up and down

- Start at the top row of the image
  - partition that row into runs of 0's and 1's
  - each run of 0's is part of the background, and is given the special background label
  - each run of 1's is given a unique component label
- For all subsequent rows
  - partition into runs
  - if a run of 1's (0's) has no run of 1's(0's) directly above it, then it is potentially a new component and is given a new label
  - if a run of 1's (0's) overlaps one or more runs on the previous row give it the minimum label of those runs
    - Let a be that minimal label and let $\{c_i\}$ be the labels of all other adjacent runs in previous row. Relabel all runs on previous row having labels in $\{c_i\}$ with a

Local relabeling

- What is the point of the last step?
  - We want the following invariant condition to hold after each row of the image is processed on the downward scan: The label assigned to the runs in the last row processed in any connected component is the **minimum** label of any run belonging to that component in the previous rows.
  - Note that this only applies to the connectivity of pixels in that part of B already processed. There may be subsequent merging of components in later rows

Example



If we did not change the c's to a's, then the rightmost a will be labeled as a c and our invariant condition will fail.

Zoran Duric

Upward scan

- A bottom to top scan will assign a unique label to each component
  - we can also compute simple properties of the components during this scan
- Start at the bottom row
  - create a table entry for each unique component label, plus one entry for the background if there are no background runs on the last row
  - Mark each component of 1's as being "inside" the background

Upward scan

- For all subsequent rows
  - if a run of 1's (0's) (say with label c) is adjacent to no run of 1's (0's) on the subsequent row, and its label is not in the table, and no other run with label c on the current row is adjacent to any run of 1's on the subsequent row, then:
    - create a table entry for this label
    - mark it as inside the run of 0's (1's) that it is adjacent to on the subsequent row
    - property values such as area, perimeter, etc. can be updated as each run is processed.
  - if a run of 1's (0's) (say, with label c) is adjacent to one or more run of 1's on the subsequent row, then it is marked with the common label of those runs, and the table properties are updated.
    - All other runs of "c's" on the current row are also given the common label.

Example

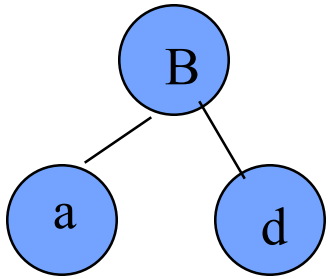
```
-------aaa
ccc---aaa
c-c---aaa
c-c---aaa
--c--aaa
aaaaaaaa
```

- changed to a during first pass
- but c's in first column will not

be changed to a's on the upward pass unless all runs are once equivalence is detected

Example

a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	D	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

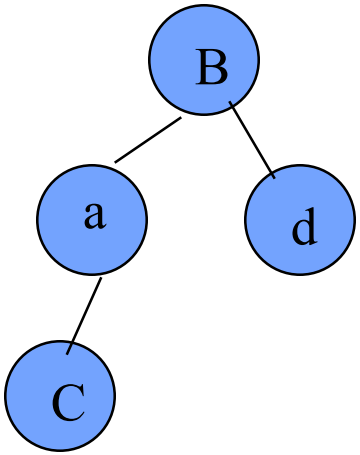


process row 4

a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

process row 3

a	a	B	b	B	B	B	B
a	a	a	a	B	c	c	c
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

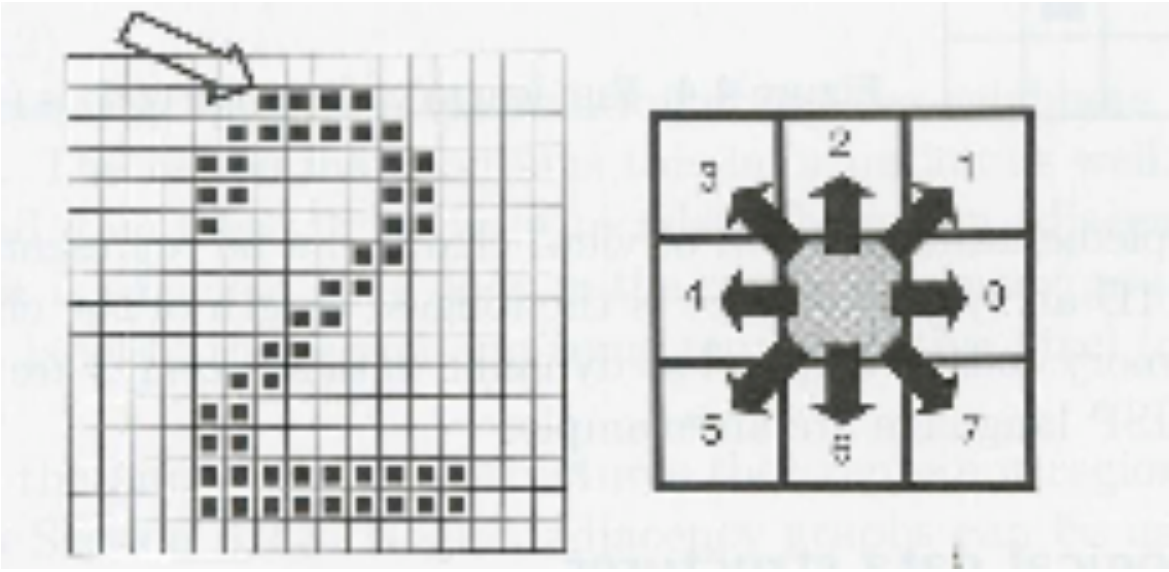


a	a	B	a	B	B	B	B
a	a	a	a	B	a	a	a
B	a	a	a	a	a	C	a
a	a	a	a	B	a	a	a
a	a	a	a	B	B	B	B
a	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d
B	a	a	a	B	d	d	d

process row 2, then 1

Chain Codes

Used for efficient boundary representation. First (reference) pixel is recorded.
All other pixels by given by their relative displacement index.



An example chain code. The reference pixel starting the chain is marked by an arrow:
0007766555555670000006444444442221111112234445652211

Integral Images

Values *ii(i,j)* at location *(i,j)* represent the sums of all the original pixel values left of and above *(i,j)*

$$ii(i,j) = \sum_{k \le i, l \le j} f(k,l)$$

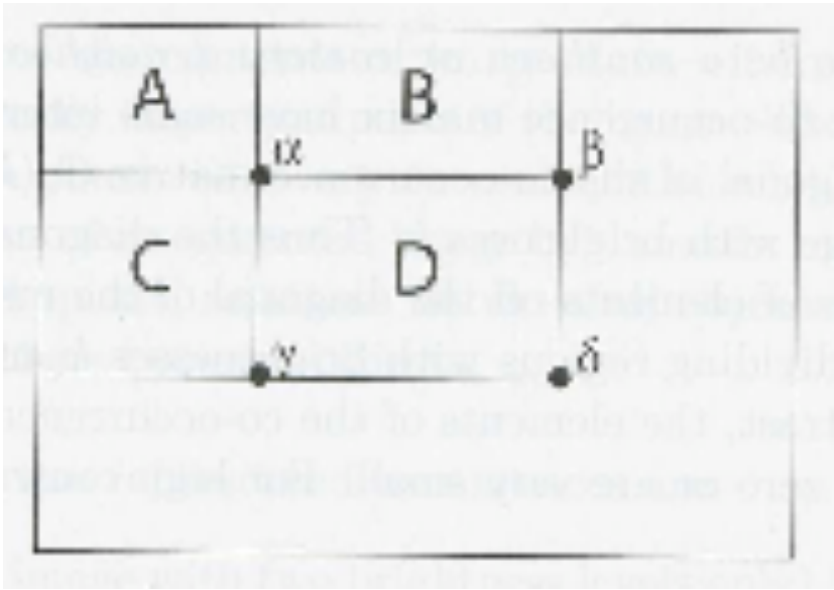
Computing Integral Images:

1. Let *s(i,j)* denote a cumulative row sum, let *s(i,-1)=0*.
2. Let *ii(i,j)* be an integral image, let *ii(-1,j)=0*.
3. Using a single row-by-row scan of the image, calculate *s(i,j)* and *ii(i,j)* using the following iterative formulas

$$s(i,j) = s(i,j-1)+f(i,j)$$

$$ii(i,j) = ii(i-1,j)+s(i,j)$$

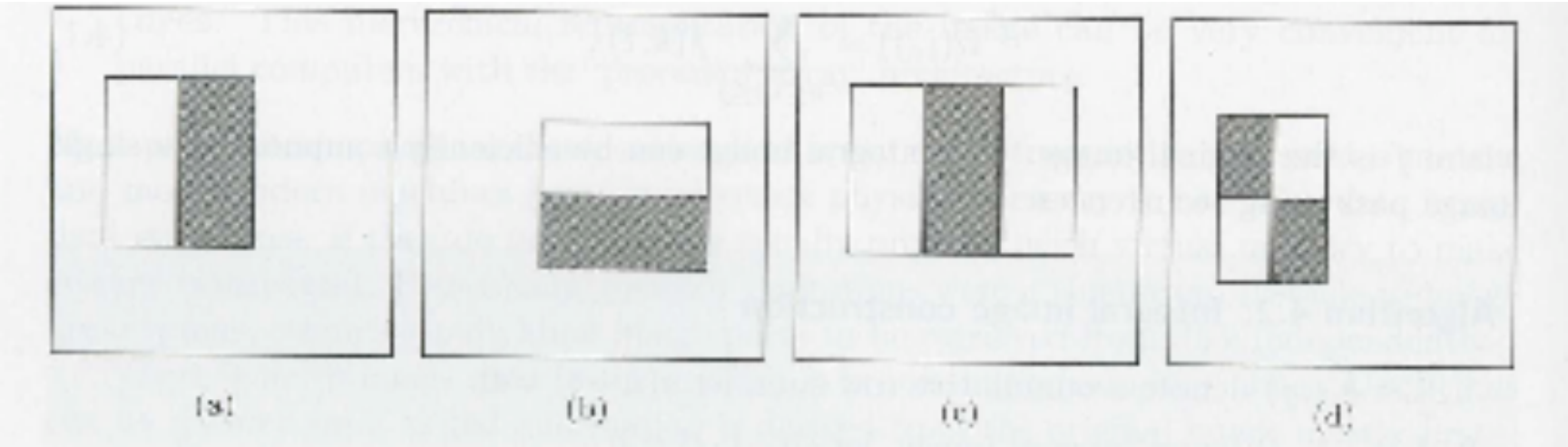
Integral Images: Computing Sums in an Area



The sum of values in area *D* can be obtained using *ii*

$$D_{sum} = ii(\delta) + ii(\alpha) - ii(\beta) - ii(\gamma)$$

Integral Images: Computing Rectangle Features



- Rectangle-based features are computed from an integral image.
- These features are computed by subtracting the sum in the shaded rectange(s) from the sum in the non-shaded rectangle(s).
- Pictures show: (a-b) two-rectangle, (c) three-rectangle, (d) four rectangle.
- These features can easily be computed at different scales/sizes.

Gray Image Histograms

Histogram *h*:

gray-level frequency distribution of the gray level image *f*

$h_f(g)$: # of pixels in *f* whose gray level is g

Cumulative histogram $H_f(g)$:

# of pixels in F whose gray level is $\leq g$

In *Matlab: imhist*

Color Histograms

- Reduced color representation =

  $C = (R/16) * 256 + (G/16)*16 + (B/16)$

  (This results in a 24 -> 12 bit color depth reduction)
- This results in a 4096 bin histogram
  - lowest 4 bits are less useful
  - requires less storage
  - faster implementation - easier to compare histograms

Color Edge Histograms

- Use edge detector to compute edges in each color band $(r_x, r_y, g_x, g_y, b_x, b_y)$
- Combine the three color bands into the structure matrix, S, to compute the color edge response
- The edge strength is computed as the larger of the two eigenvalues of S, and the orientation is given by the corresponding eigenvector
- Histogram bin index is determined using edge orientation (36 bins total), and the bin count is incremented using the edge magnitude

Histogram Matching

- Histogram Intersection

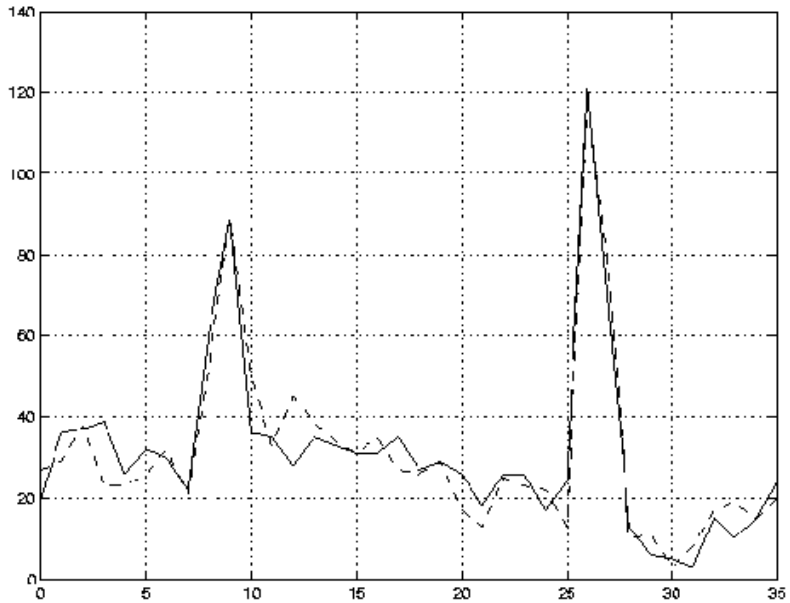
$$I(h_c, h_b) = \frac{\sum_i \min\{h_c(i), h_b(i)\}}{\sum_i \max\{h_c(i), h_b(i)\}}$$

- Chi Squared Formula

$$\chi^2(h_c, h_b) = \sum_i 2\frac{(h_c(i) - h_b(i))^2}{h_c(i) + h_b(i)}$$

Examples of edge histograms

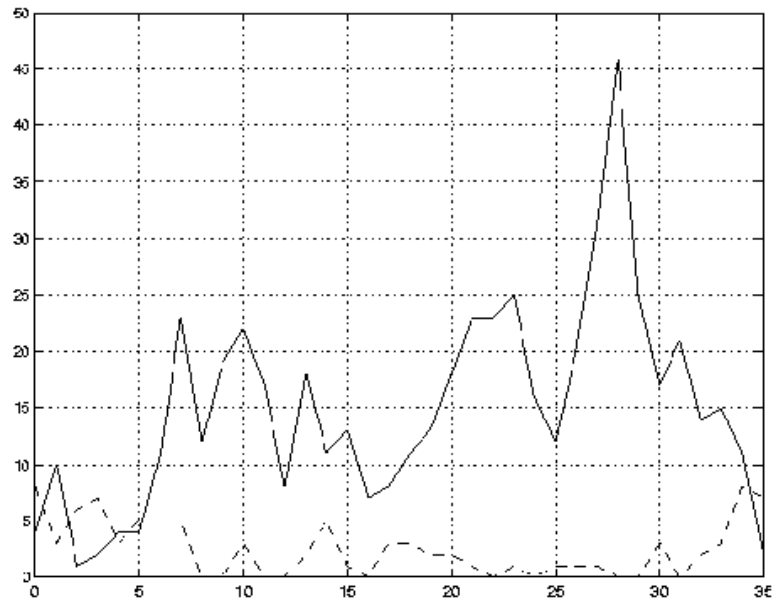
similar histograms



Similarity (inters.) = 92%

$X^2 = 61$

different histograms



Similarity (inters.) = 22%

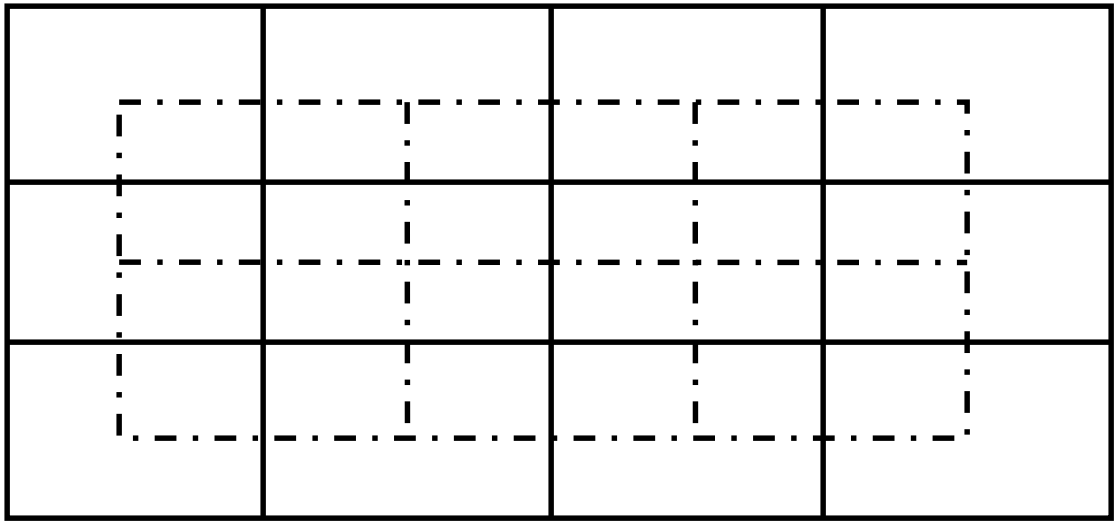
$X^2 = 828$

Using Histograms for Background Modeling

- Use histograms of small regions to model the background:
  - Color histograms computed for small regions of the "background" image and the current (new) image (reduced color/ 12 bit bit representation)
  - Color edge histograms computed for small regions of the "background" image and the current image (36 bin quantization)

Overall Control

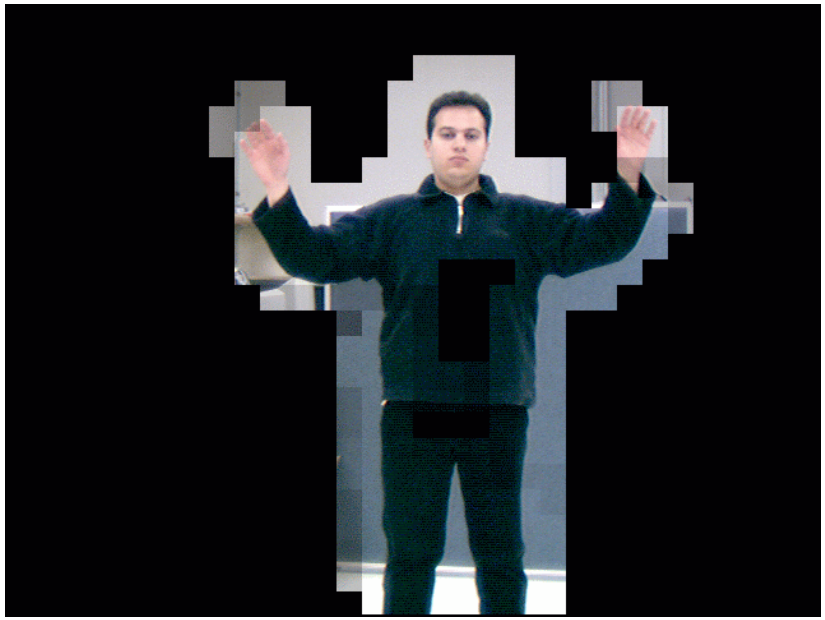
- Divide each frame into *40x40* pixel blocks
- To make sure that we do not miss objects on grid block boundaries we tile the frame by overlaying two grids, one of which is shifted by *20* pixels in *x* and *y* directions



Criteria for Block Activation

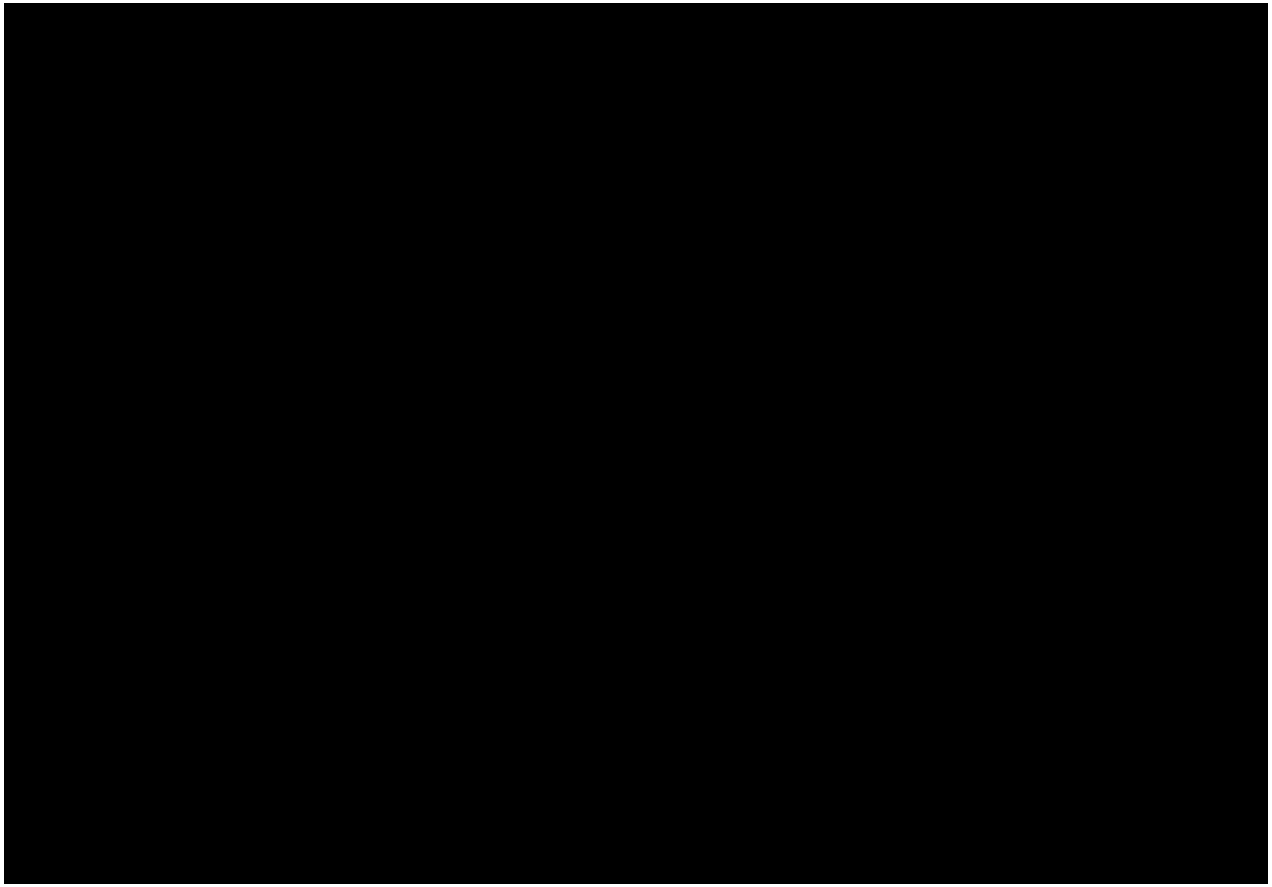
- On a block by block basis, similarity measures between background and foreground histograms are computed
- For histogram intersection: If the similarity is below a threshold, $T_{\cap}$, then the block contains a foreground object and is activated for display
- For chi squared: If the X^2 measure is greater than a threshold, T_{χ}, then the block contains a foreground object and is activated for display

Using Edge Histograms for Detection



Moving Person in a Cluttered Scene

Color Histogram Based Detection



Edge Histogram-Based Detection

Surveillance: Dropping an Object

Surveillance: Removing an Object

Surveillance: Interacting People