Workshops in Creative Computing: Computer Vision Module

Lecture 3: Blob Detection

Wednesday March 6, 2013

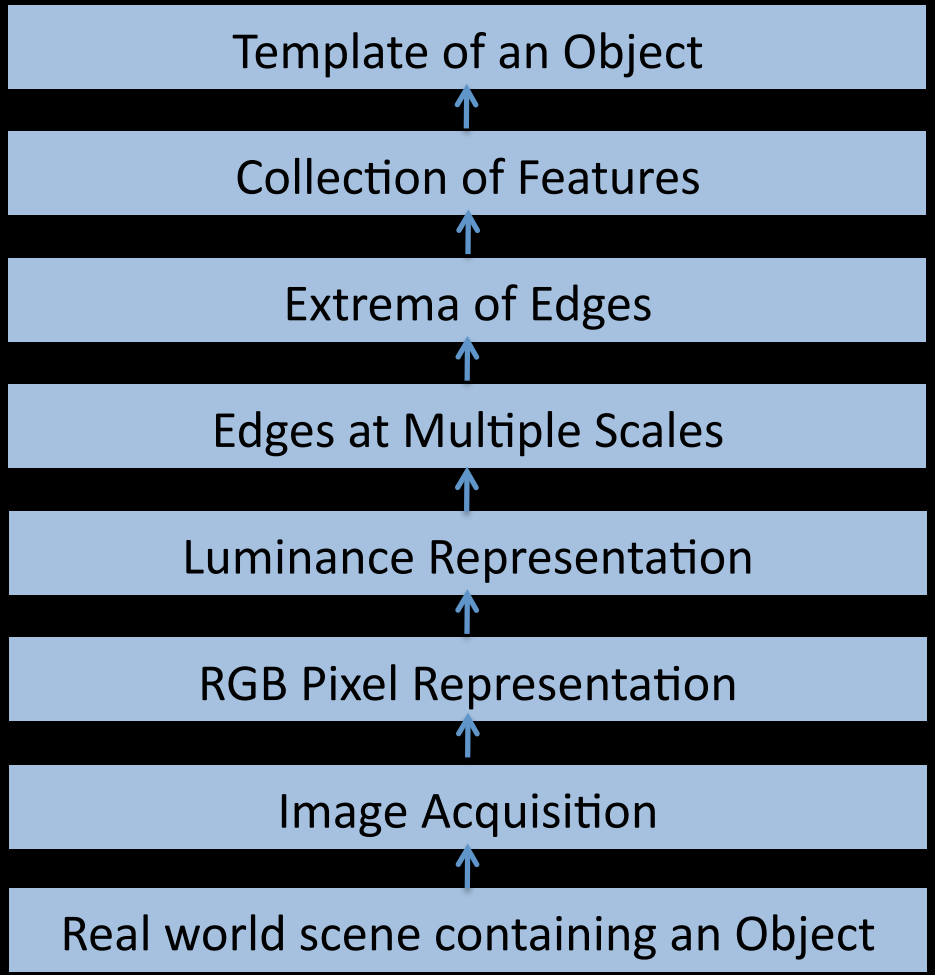
Parag K Mital

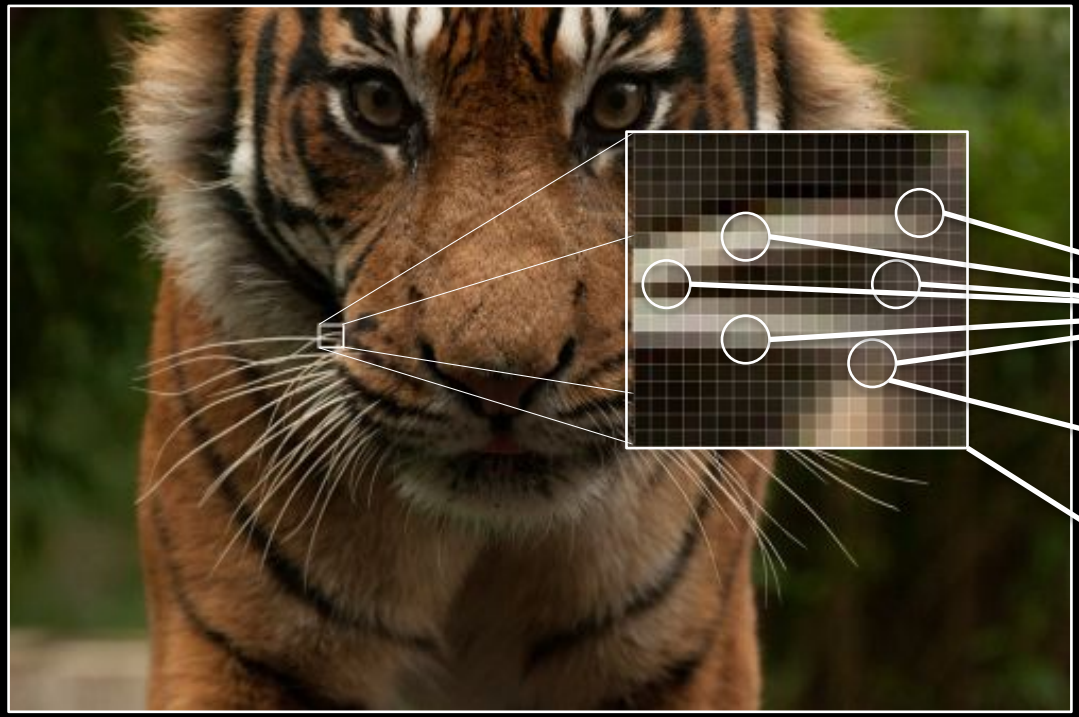
Did we solve Computer Vision?

Vision is based on inference

What makes us perceive objects in images?

Hypothesis: *process images bottom-up*
– Extract “features”
– Combine features with prior knowledge to classify objects in the image at a high-level


Template of an Object
Collection of Features
Extrema of Edges
Edges at Multiple Scales
Luminance Representation
RGB Pixel Representation
Image Acquisition
Real world scene containing an Object



Semantic label =
High-level description

Grouping of Features =
Mid-level description

Single feature =
Low-level description

Pixels =
Low-level description

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

– Store background

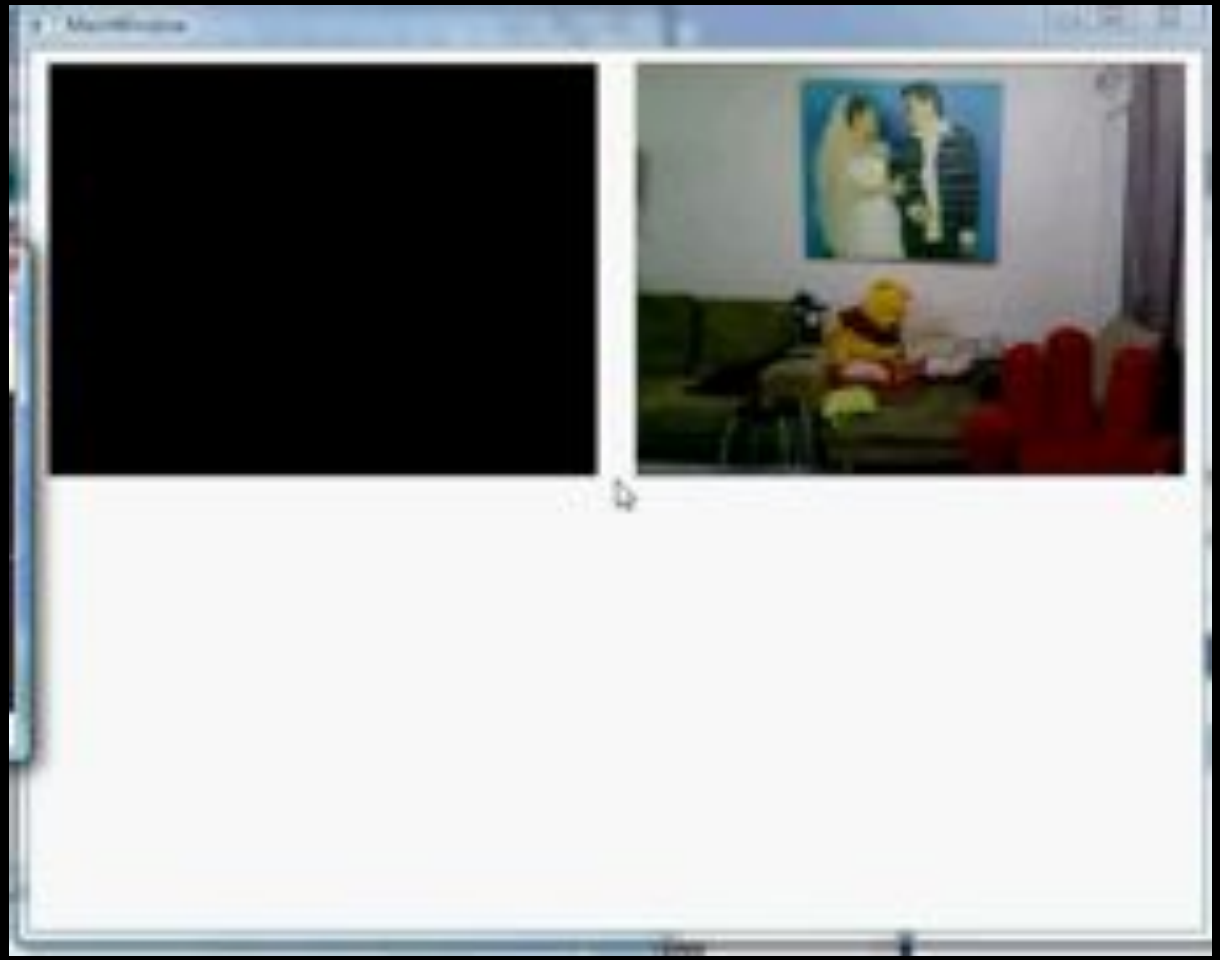
– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

– Collect foreground pixels into blobs

– Track blobs into the next frame

Ideal setup for detecting Blobs

Infrared Camera
Depth Sensor
High Contrast Background
Controlled Lighting
- Infrared spots for infrared
- No-Infrared for Kinect



Popular Uses of Blob Detection:

Shape Detection (e.g. Convex Hull)
Motion Capture (e.g. Pfinder)
Human-Computer Interaction (e.g. Kreuger)
Action Detection (e.g. Anomalous activities)
Surveillance (e.g. Tracking Humans/Cars/
Traffic Analysis)


Funky Forest
An interactive ecosystem


Pfinder
Real-Time Tracking of the Human Body

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

– Store background

– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

– Collect foreground pixels into blobs

– Track blobs into the next frame

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

– Store background

– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

– Collect foreground pixels into blobs

– Track blobs into the next frame

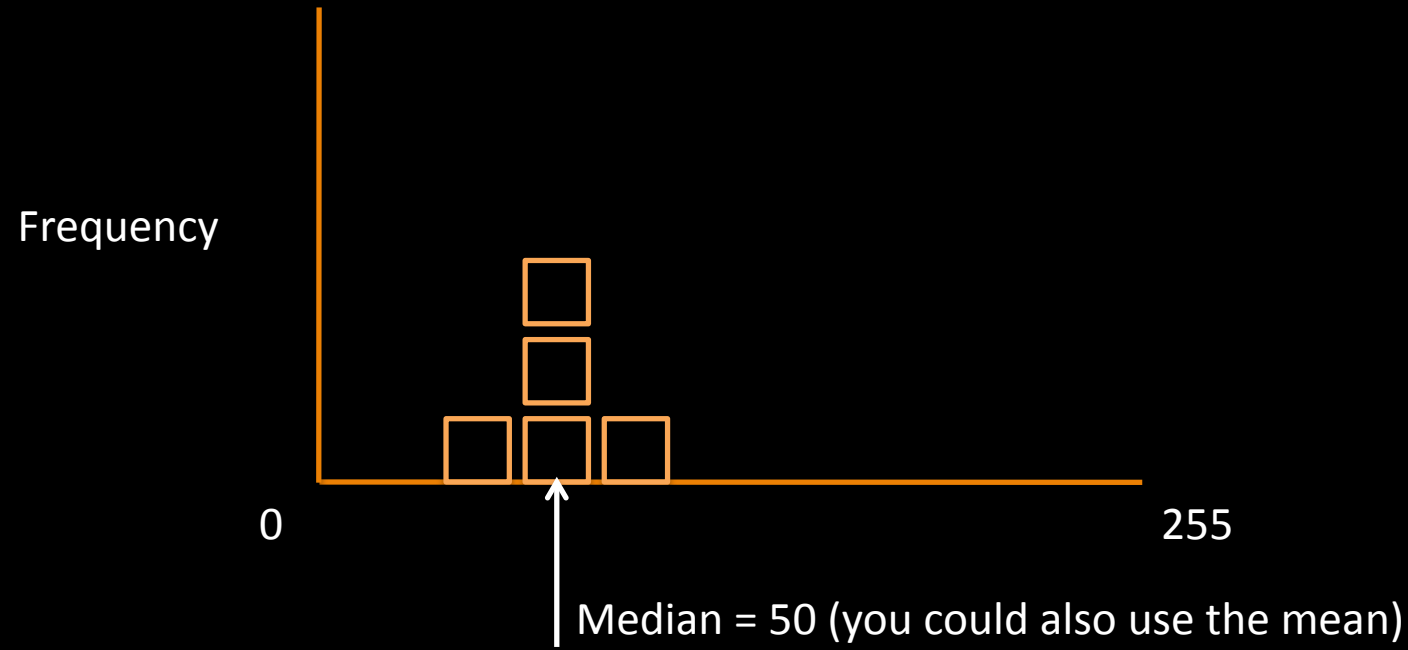
Easy: take a picture without anyone/anything in it

Harder: Unable to remove all foreground objects…

Ideas?


Frame: 1


Frame: 2
Frame: 3
Frame: 4
Frame: 5


Frequency
0
255
Median = 50 (you could also use the mean)

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

– Store background

– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

– Collect foreground pixels into blobs

– Track blobs into the next frame

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

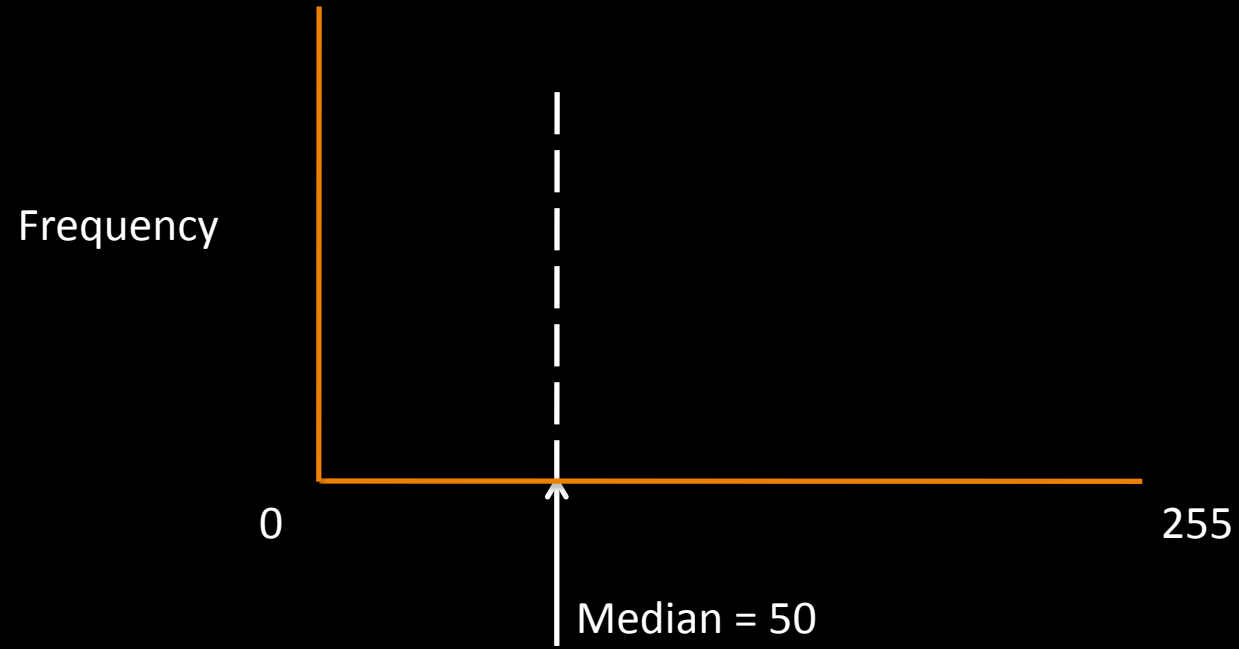
~~– Store background~~

– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

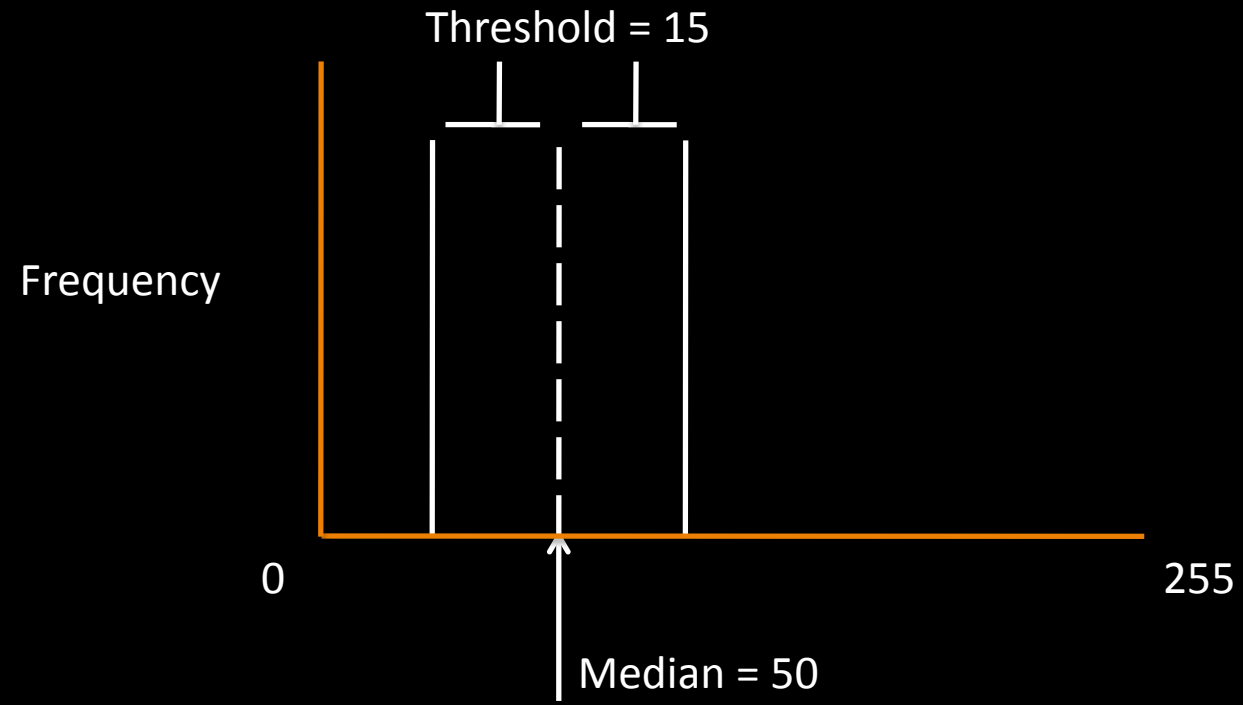
– Collect foreground pixels into blobs

– Track blobs into the next frame

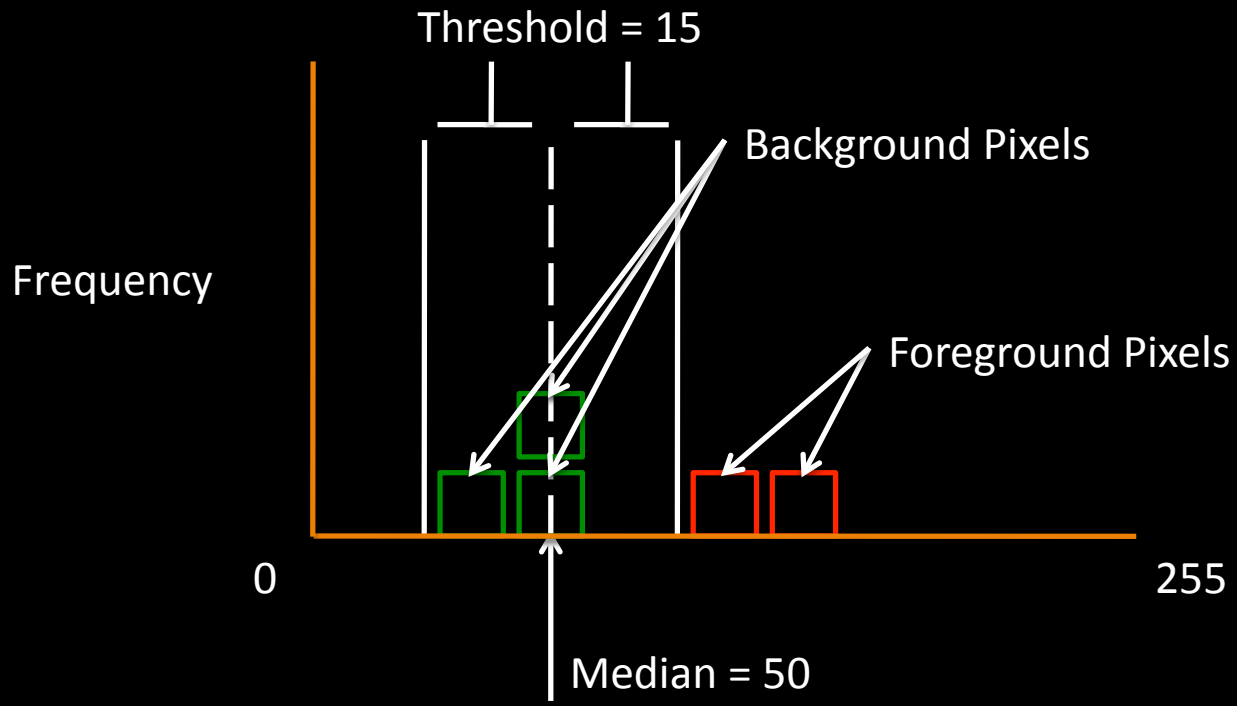

Frame: 2
Frame: 3
Frame: 4
Frame: 5


Frequency
0
255
Median = 50

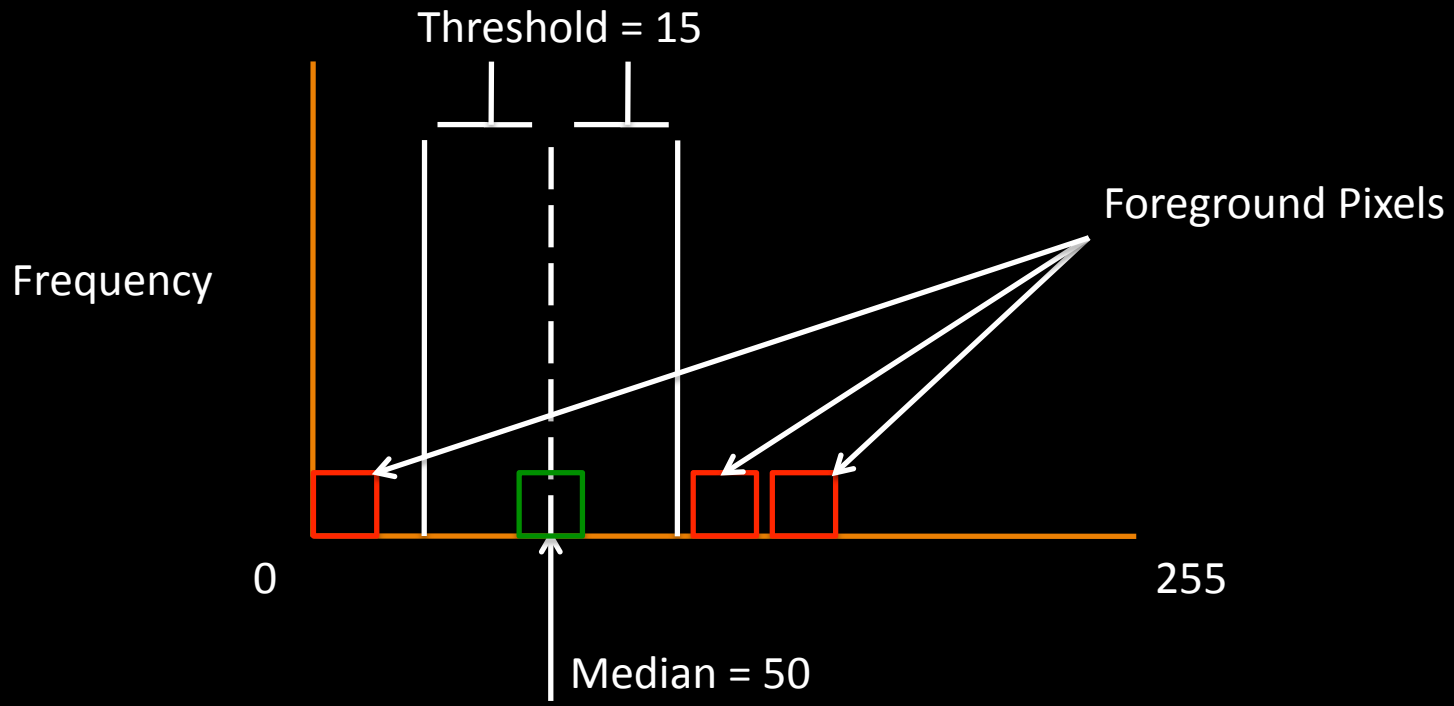

Frame: 2
Frame: 3
Frame: 4
Frame: 5


Threshold = 15
Frequency
0
255
Median = 50


Frame: 2
Frame: 3
Frame: 4
Frame: 5


Threshold = 15
Background Pixels
Frequency
Foreground Pixels
0
255
Median = 50

Background Subtraction



Background Subtraction

Issues:

1. Fixed model doesn't interpret changes in lighting over time.

2. What is a good threshold

3. What if the background isn't described by a single median value, but 2, or 3…, and those medians change over time?

Background Subtraction

Issues:

1. Fixed model doesn't interpret global changes in scene over time.

2. What is a good threshold

3. What if the background isn't described by a single median value, but 2, or 3…, and those medians change over time?

Use a *running average* of the past X seconds of observations to train our background

Use a *probabilistic density model* such as a Gaussian Model

Use a *mixture* of Gaussian models

GMM Background Model

1. For each pixel, add the previous $\mathbb{X}$ seconds worth of observations to a vector
2. Train a Gaussian Mixture Model on all observations
3. A foreground pixel is any pixel not explained by the GMM

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

~~– Store background~~

– Anything unlike background becomes “figure”, or foreground, perform operation per pixel

– Collect foreground pixels into blobs

– Track blobs into the next frame

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

~~– Store background~~

~~– Anything unlike background becomes "figure", or foreground, perform operation per pixel~~

– Collect foreground pixels into blobs

– Track blobs into the next frame

Connected Components

On the first pass:

Iterate through each element of the data by column, then by row (Raster Scanning)

If the element is not the background

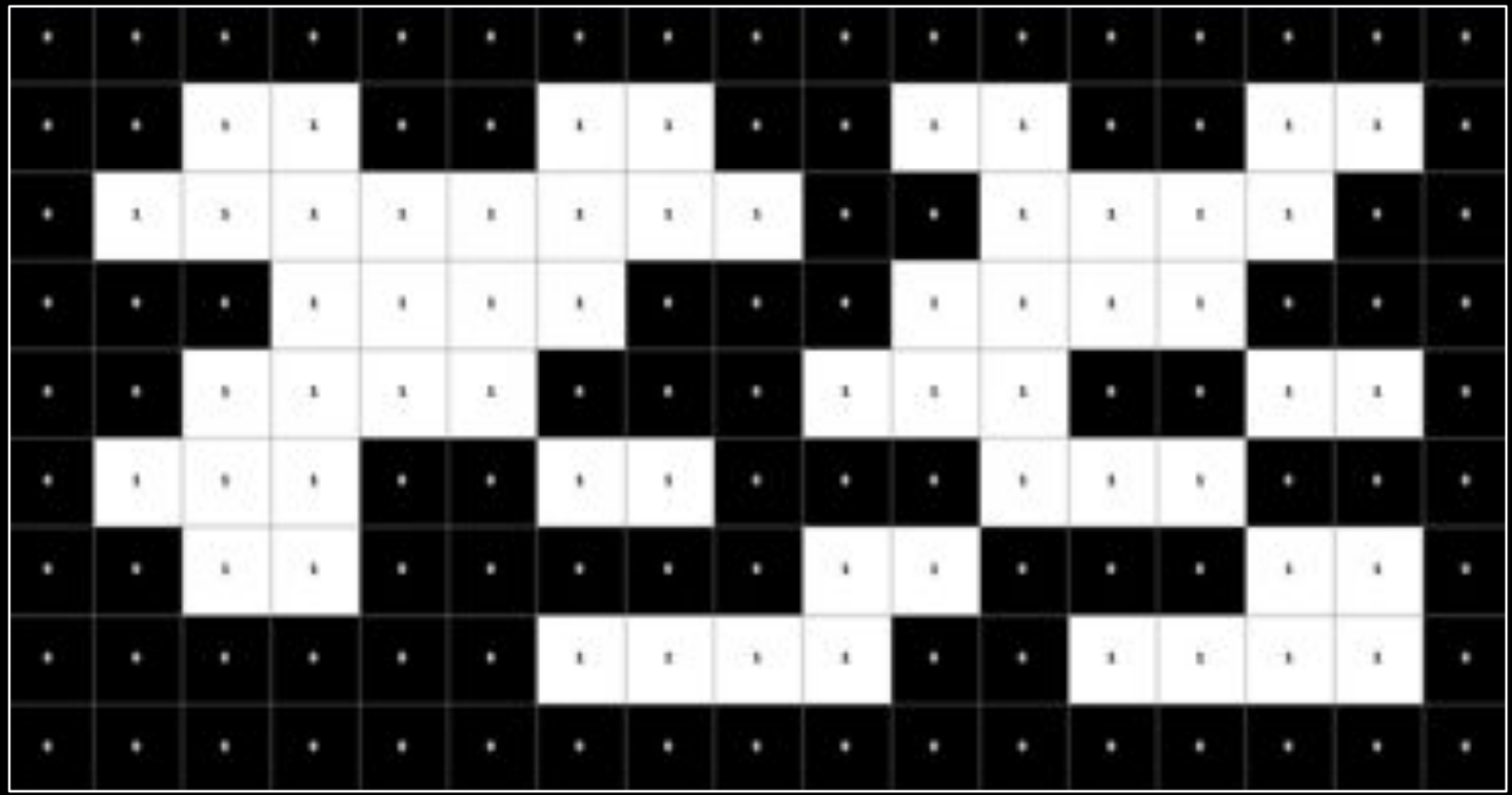
- Get the neighboring elements of the current element
- If there are no neighbors, uniquely label the current element and continue
- Otherwise, find the neighbor with the smallest label and assign it to the current element
- Store the equivalence between neighboring labels

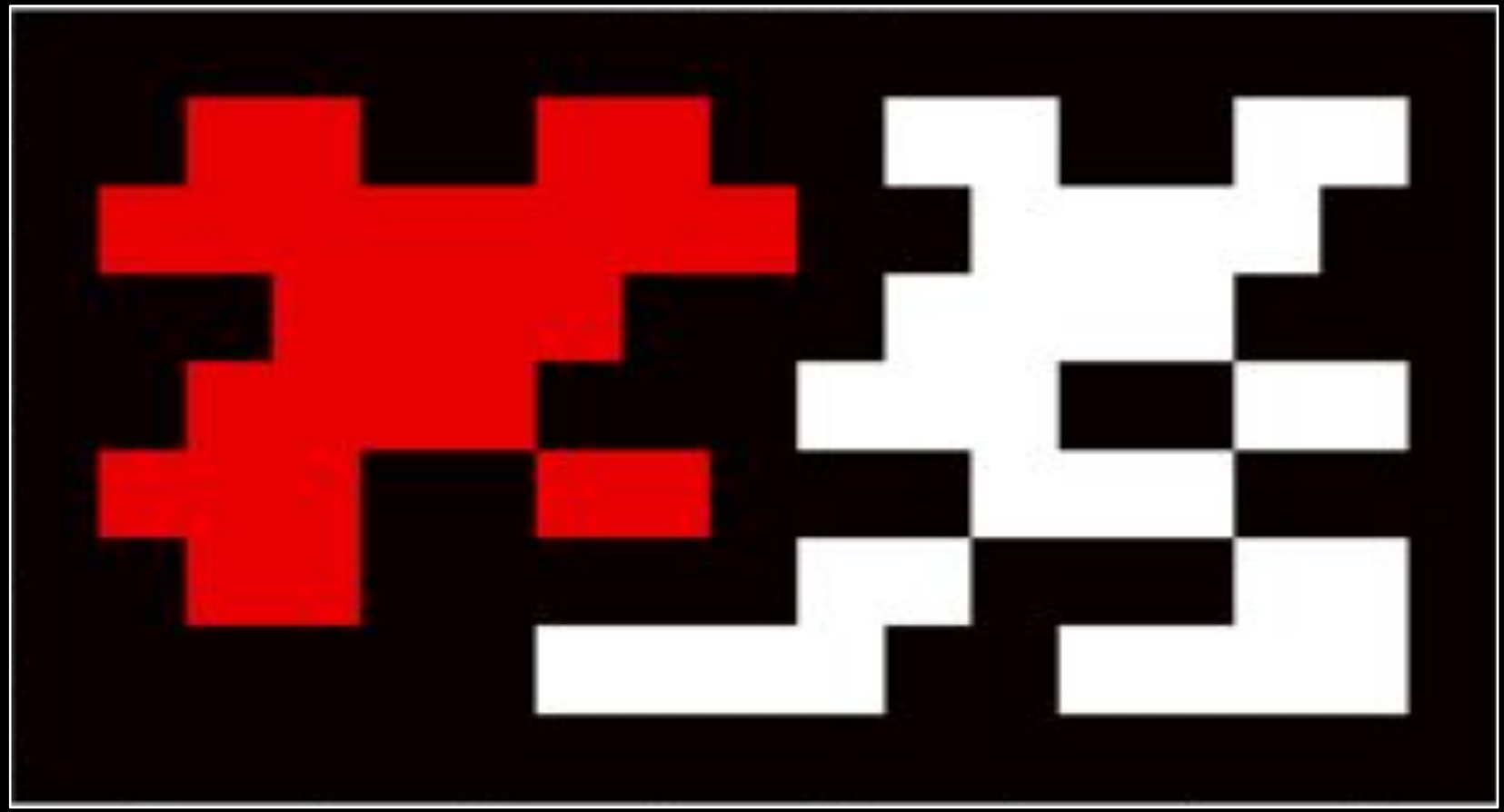
On the second pass:

Iterate through each element of the data by column, then by row

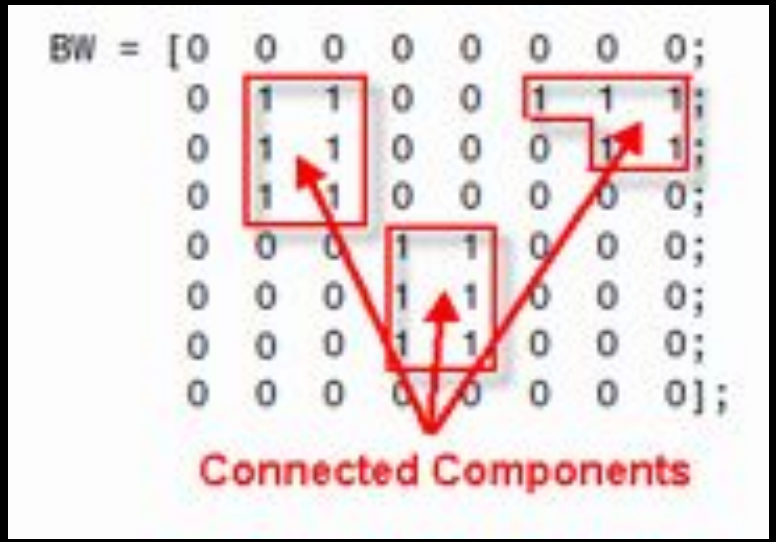
If the element is not the background

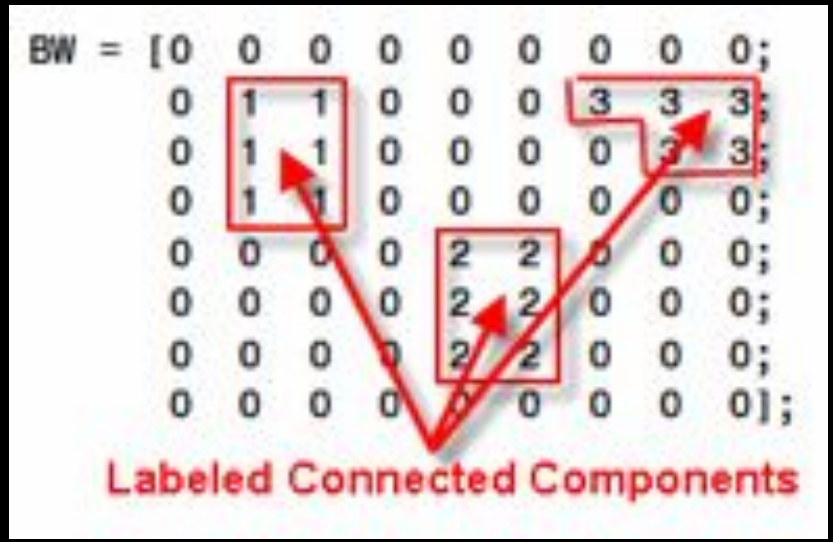
- Relabel the element with the lowest equivalent label





Connected Components





Connected Components

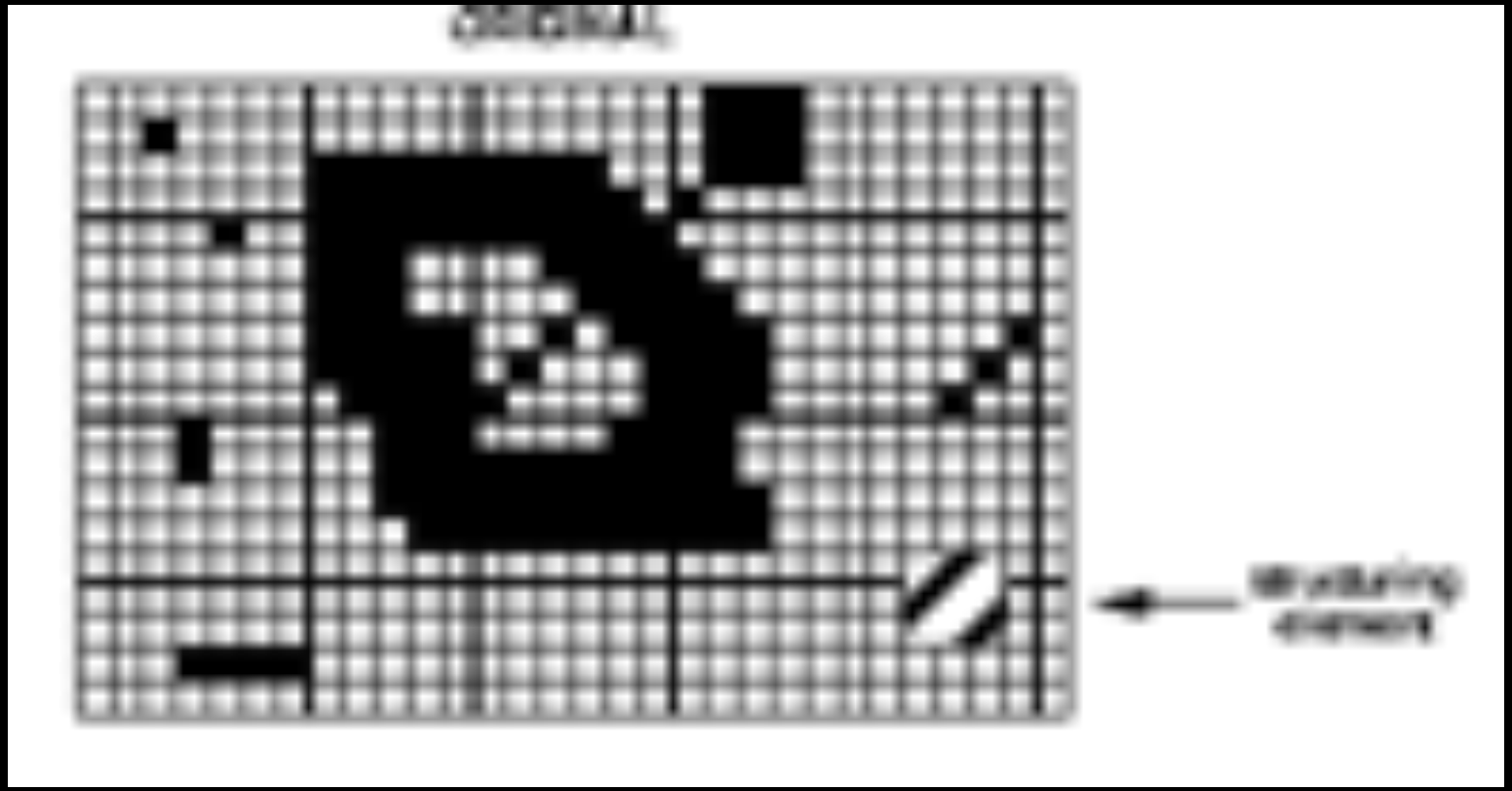
Only works on binary images (thresholded images)

Collects connected regions into labeled groups

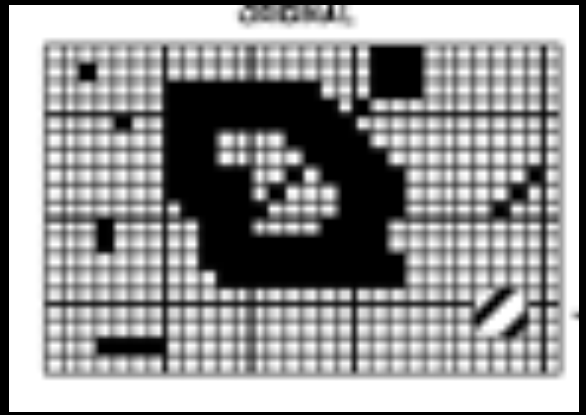
Useful if you have multiple objects to track

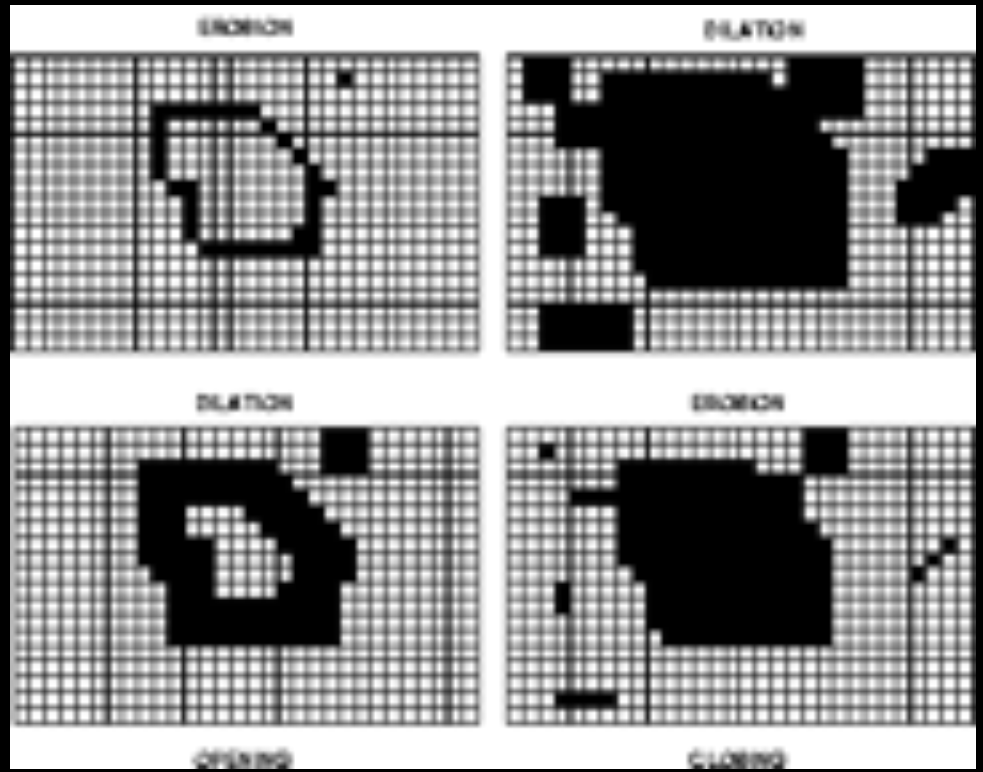
Doesn’t work well with noisy images (poor thresholding)

Morphological Operators



Morphological Operators





Morphological Operators

Dilation

The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.

Erosion

The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

Morphological Operators

Dilation

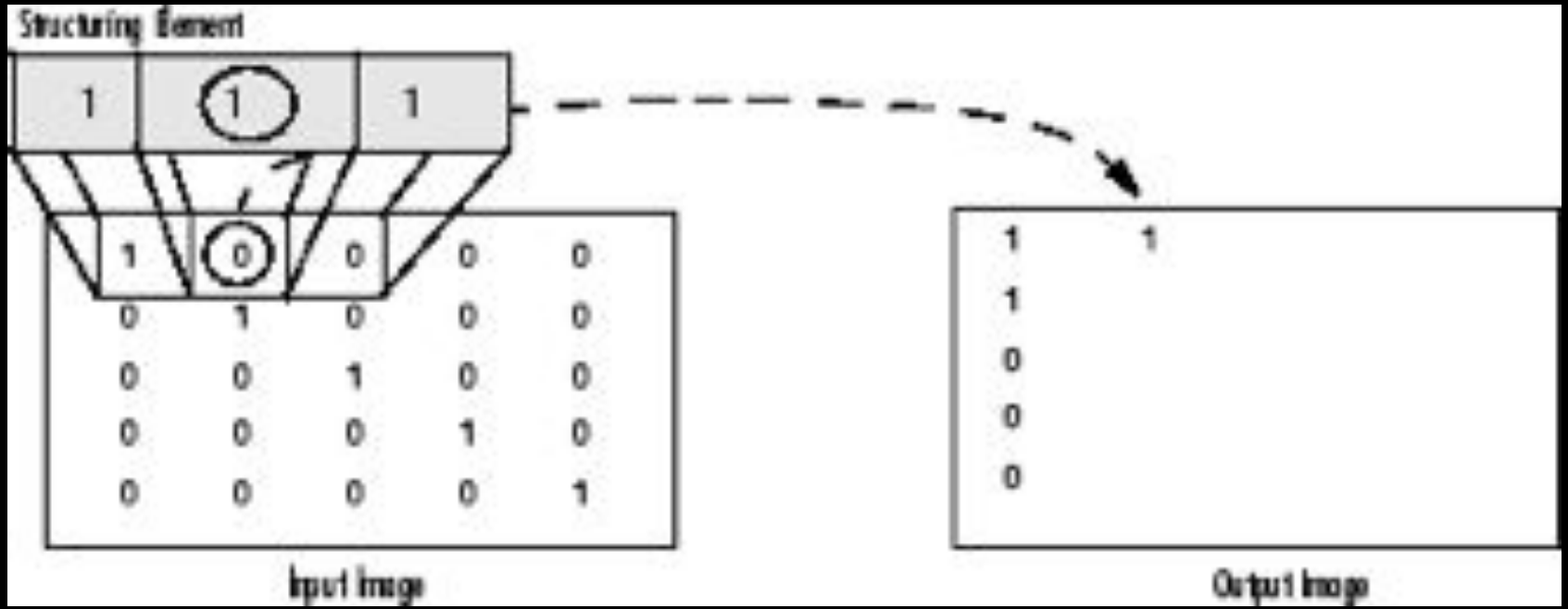
Adds pixels to the boundaries of objects in an image

Erosion

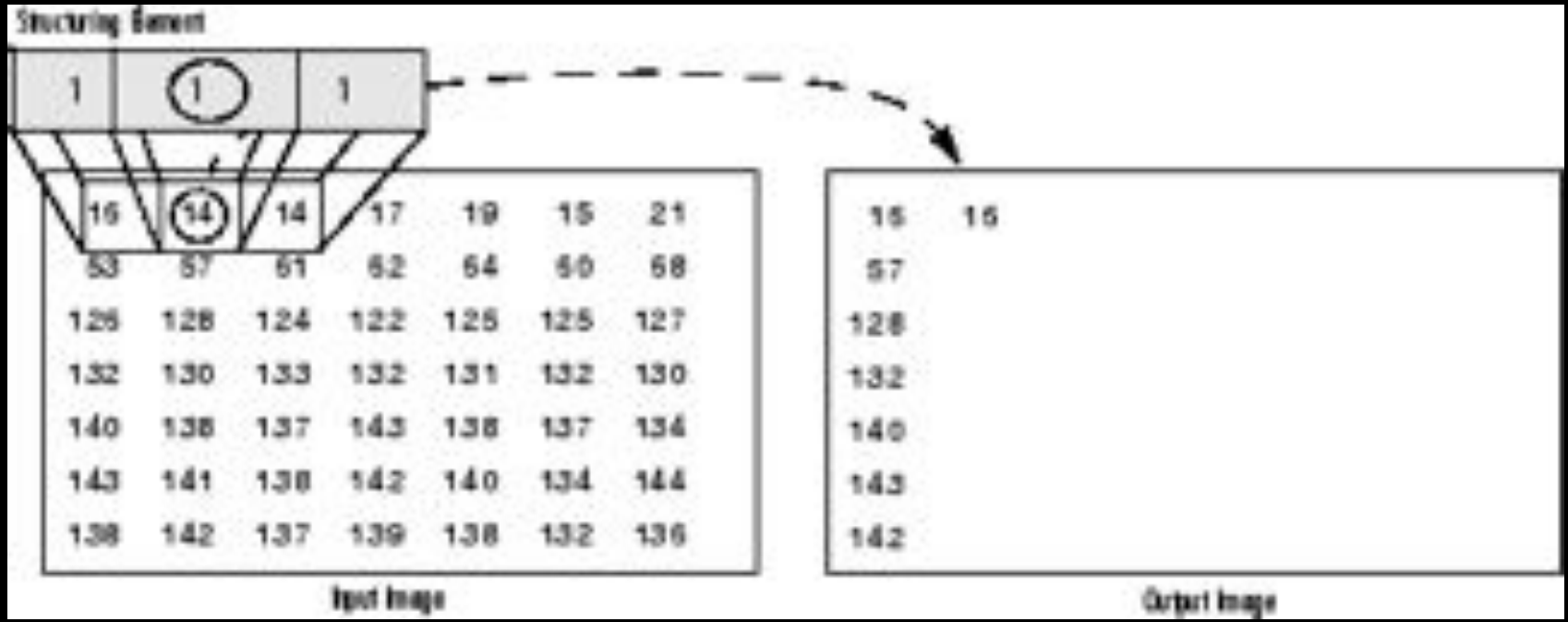
Removes pixels on object boundaries

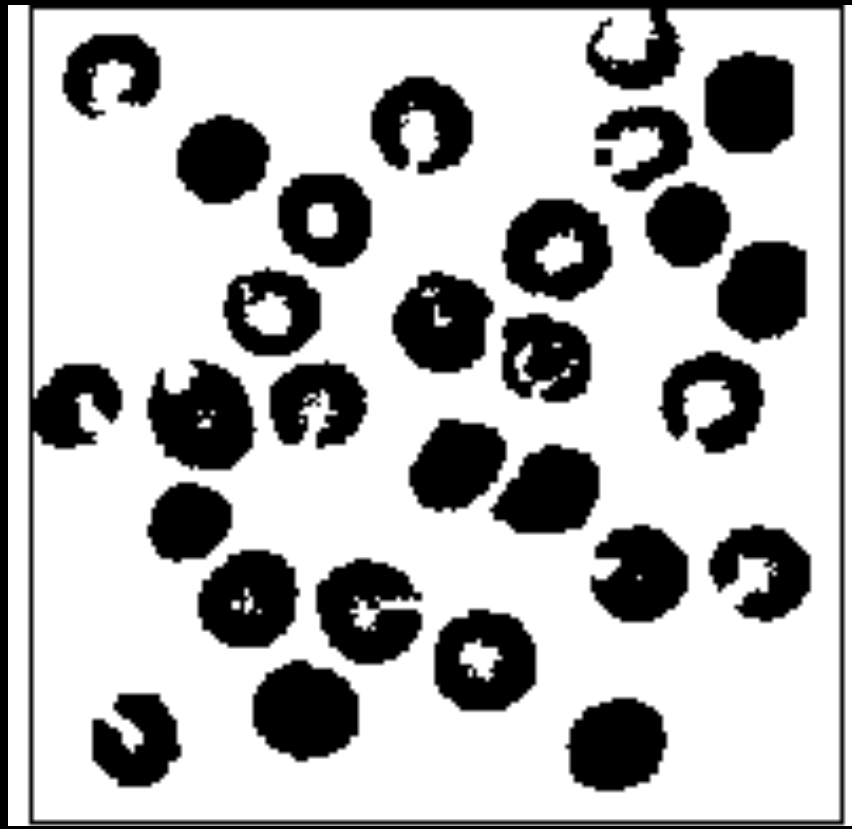
The number of pixels added or removed from the objects in an image depends on the size and shape of the *structuring element* used to process the image.

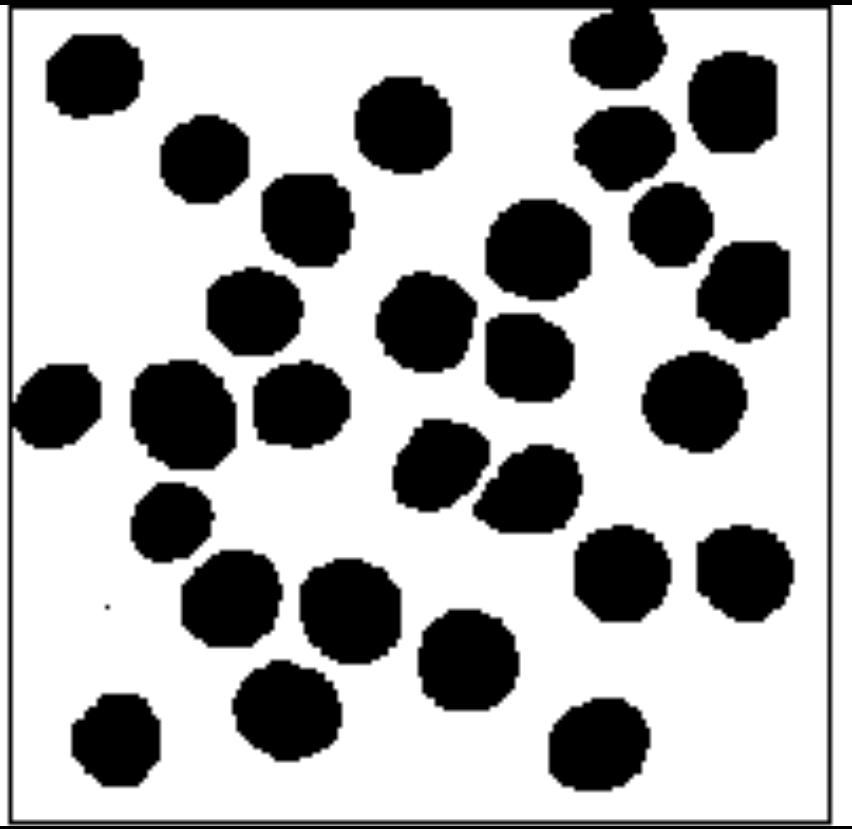
Morphological Operators



Morphological Operators








['space'] to learn background
['+'] or ['-'] to change threshold: 50
['9'] or ['0'] to change gmm timing: 40.000000
['['] or [']'] to change block size: 13

Morphological Operators

Can help you get clean blobs
Removes noise
Closes holes
Use in conjunction with blurring
Apply *before* connected components analysis

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

~~– Store background~~

~~– Anything unlike background becomes "figure", or foreground, perform operation per pixel~~

– Collect foreground pixels into blobs

– Track blobs into the next frame

What makes us perceive objects in 3D?

Hypothesis: *things separate from ground must belong to objects (figure/ground; gestalt grouping)*

~~– Store background~~

~~– Anything unlike background becomes "figure", or foreground, perform operation per pixel~~

~~– Collect foreground pixels into blobs~~

– Track blobs into the next frame

Recognition ≠ Detection ≠ Tracking

Tracking blobs

Problem Statement:

Multiple blobs can be found each frame. How do we keep a unique id for each blob that is tracked over time, for each new frame, and which disappears when the blob leaves?

Can we infer the blob's id again if it comes back into view after having disappeared?

Can we infer where the blob *will* go, extrapolating their movement into the future?

Tracking blobs

Simplest algorithm:

1. Assign each blob a unique id.
2. Sort the distances to each blob.
3. Use the blob id with the smallest distance.

Tracking blobs

Simplest algorithm:

1. Assign each blob a unique id.
2. Sort the distances to each blob.
3. Use the blob id with the smallest distance.

Issues:

• Doesn't know what to do when blobs collide to become one blob

• Can't guess into the future (though can w/ minimal work)

Tracking blobs

Other popular algorithms:

1. Kalman Filter (e.g. GPS)
2. Particle Filter
3. Conditional Density Propagation (Condensation) Tracking
4. Appearance-based methods

Possible Blob Features

Area
Perimeter
Contour
Centroid
Bounding Box
Orientation
Velocity
Acceleration
Silhouette
Shape (Convex Hull)
Skeleton
Movement? Behavior? Gesture?

Control your environment

Stable Camera
Minimal external lighting
Manual exposure
Manual focus
Camera resolution
Camera framerate

Possible Directions

- Appropriation of CCTV footage
- Detecting shape of Blobs for doing more interesting interaction than just the position, e.g. Kreuger/Funky Forest
- Live video mixing with silhouette