Computer Vision

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

University of Bucharest, 2nd semester, 2023-2024

Course structure

1. **Features and filters: low-level vision**

   Linear filters, color, texture, edge detection, template matching

2. **Grouping and fitting: mid-level vision**

   Fitting curves and lines, robust fitting, RANSAC, Hough transform, segmentation

3. **Multiple views**

   Local invariant feature and description, epipolar geometry and stereo, object instance recognition

4. **Object Recognition: high – level vision**

   Object classification, object detection, part based models, bovw models

5. **Video understanding**

   Object tracking, background subtraction, motion descriptors, optical flow

Background subtraction

- given an image (video frame) we want to identify the ***foreground objects*** in that image.
- in most cases, objects are of interest and not the scene

$\Rightarrow$

Slide credit: Birgi Tamersoy

Background subtraction

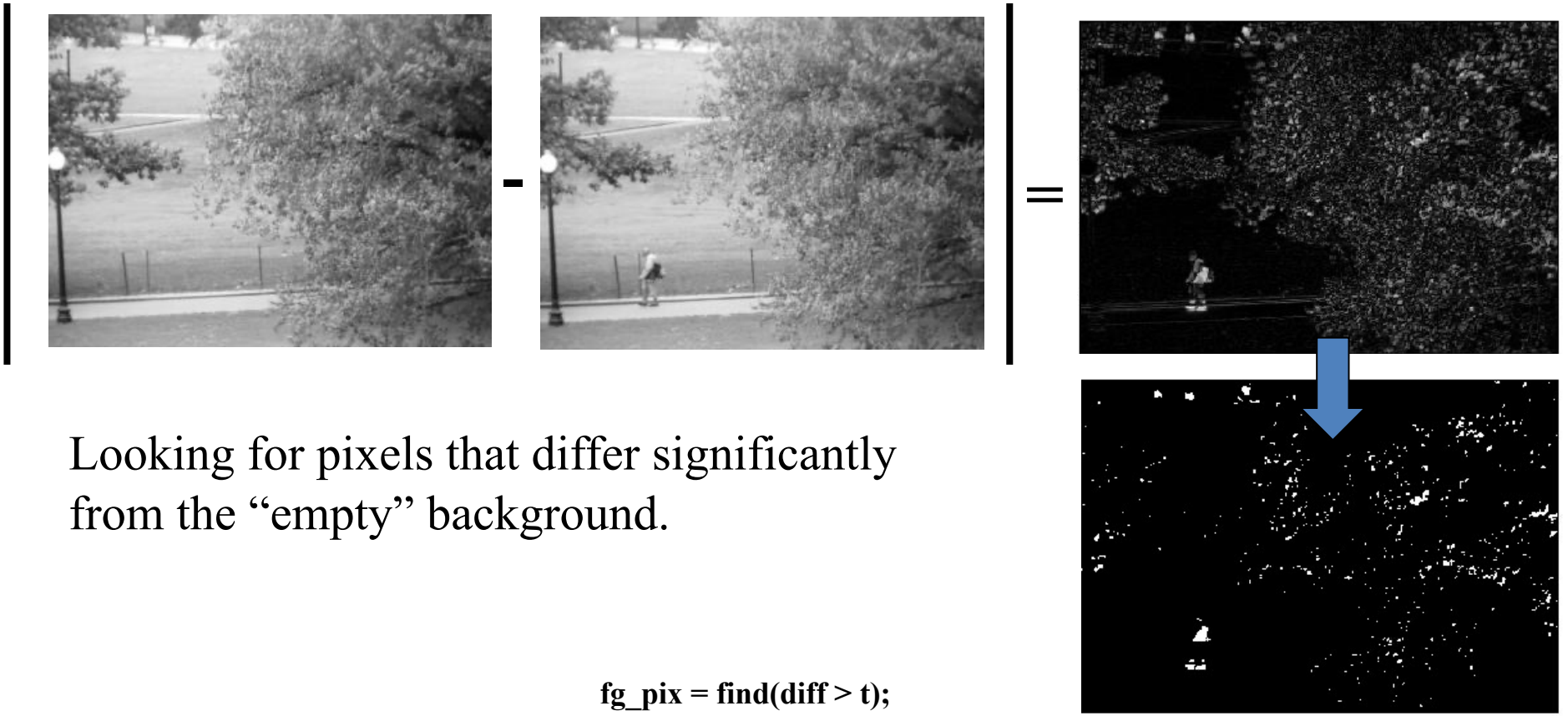
- simple techniques can do ok with static camera
- …but hard to do perfectly

- widely used:
  - traffic monitoring (counting vehicles, detecting & tracking vehicles, pedestrians),
  - human action recognition (run, walk, jump, squat),
  - human-computer interaction
  - object tracking

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: background subtraction



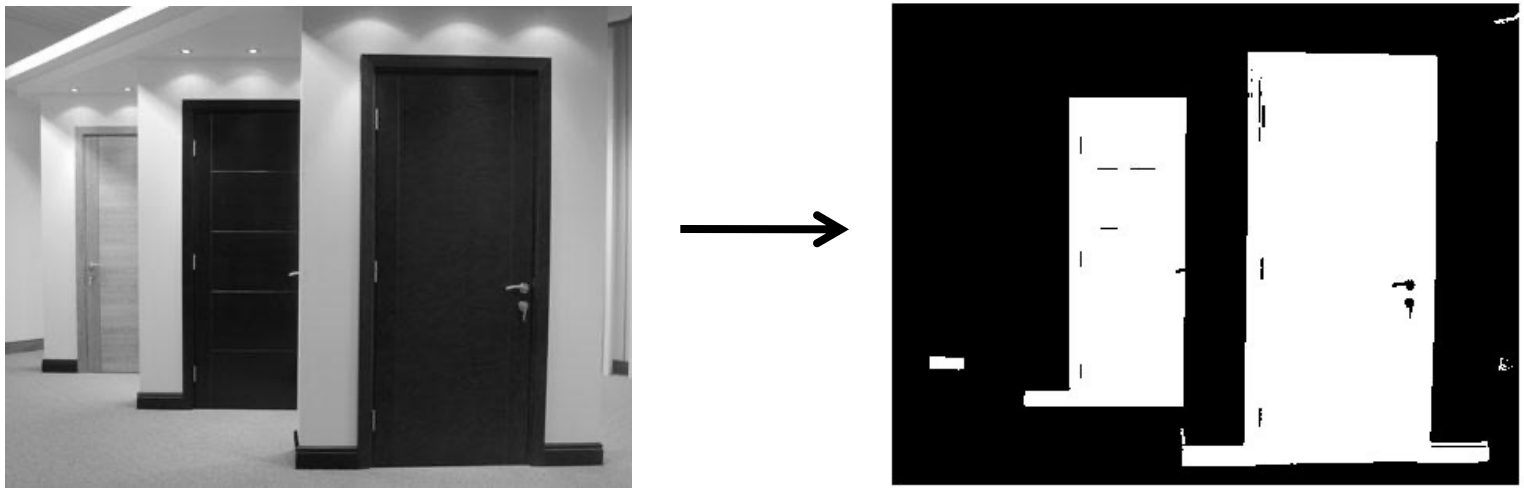
Looking for pixels that differ significantly from the "empty" background.

fg_pix = find(diff > t);

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: intensity-based detection



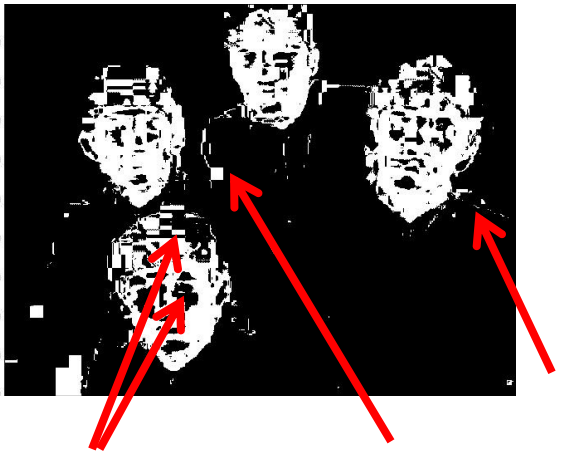
fg_pix = find(im < 65);

Looking for dark pixels

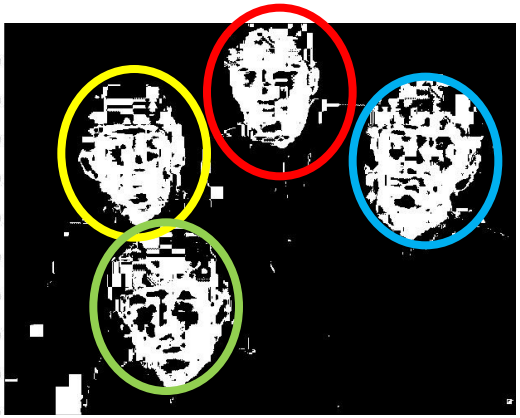
Slide credit: Kristen Grauman

Issues

- What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments



- How to delimitate multiple regions of interest?
  - Count objects
  - Compute further features per object



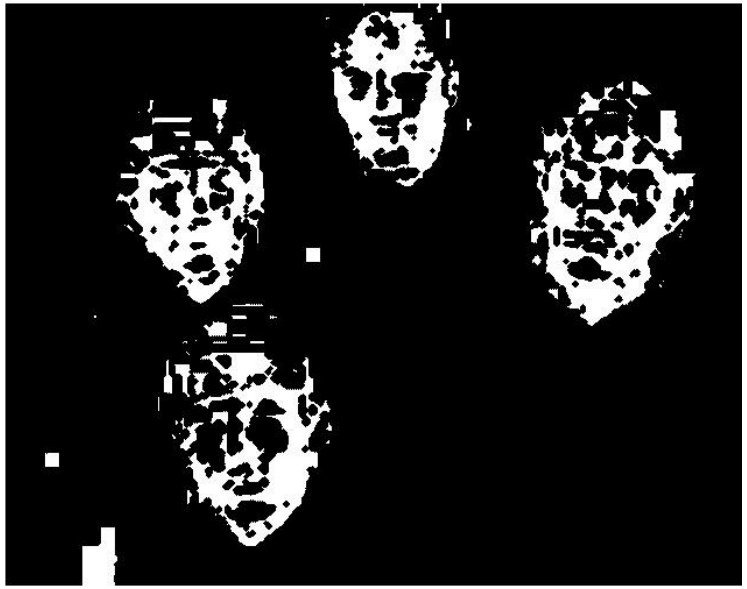
Slide credit: Kristen Grauman

Morphological operators

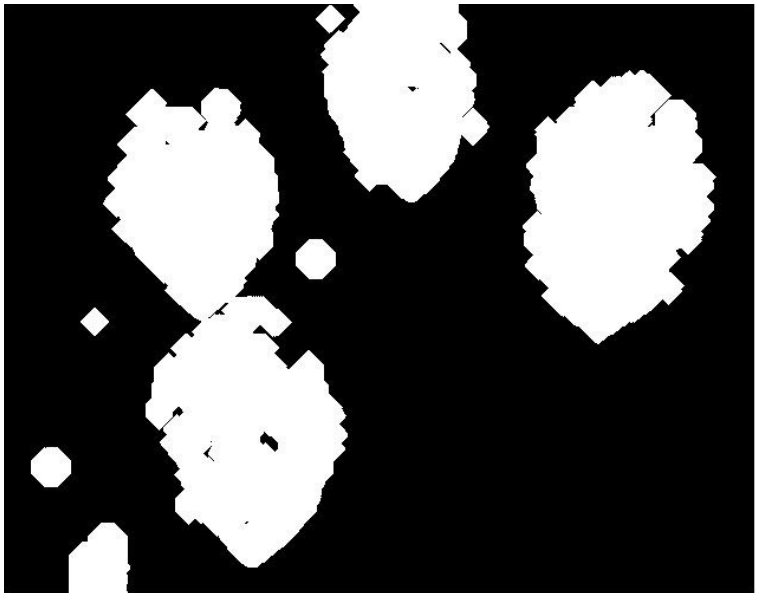
- Change the shape of the foreground regions via intersection/union operations between a scanning structuring element and binary image.
- Useful to clean up result from thresholding
- Basic operators are:
  - Dilation
  - Erosion

Dilation

- Expands connected components
- Grow features
- Fill holes



Before dilation



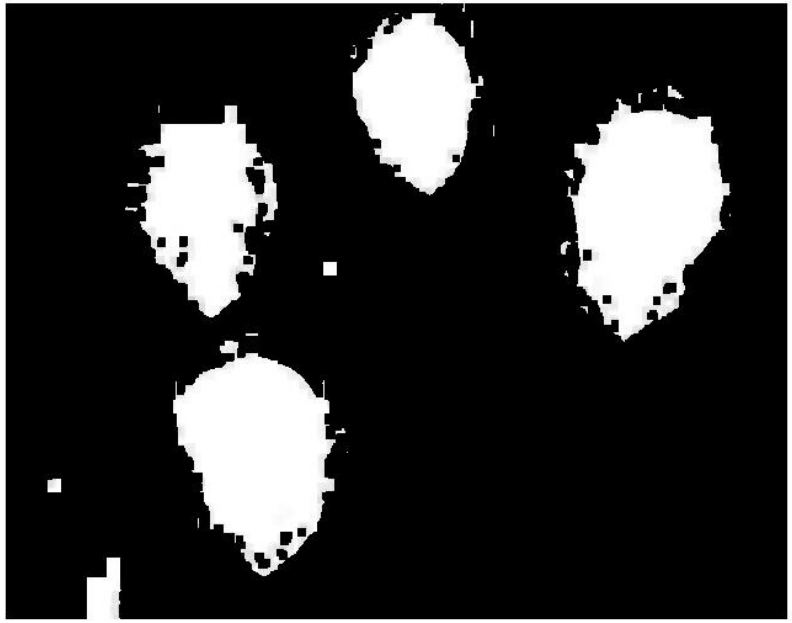
After dilation

Slide credit: Kristen Grauman

Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise

Before erosion

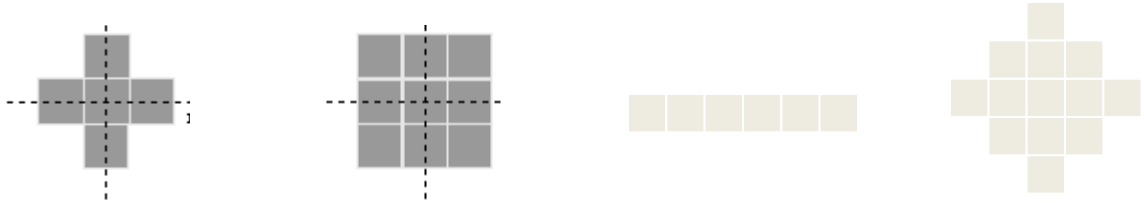


After erosion

Slide credit: Kristen Grauman

Structuring elements

- **Masks** of varying shapes and sizes used to perform morphology, for example:



- Scan mask across foreground pixels to transform the binary image

Slide credit: Kristen Grauman

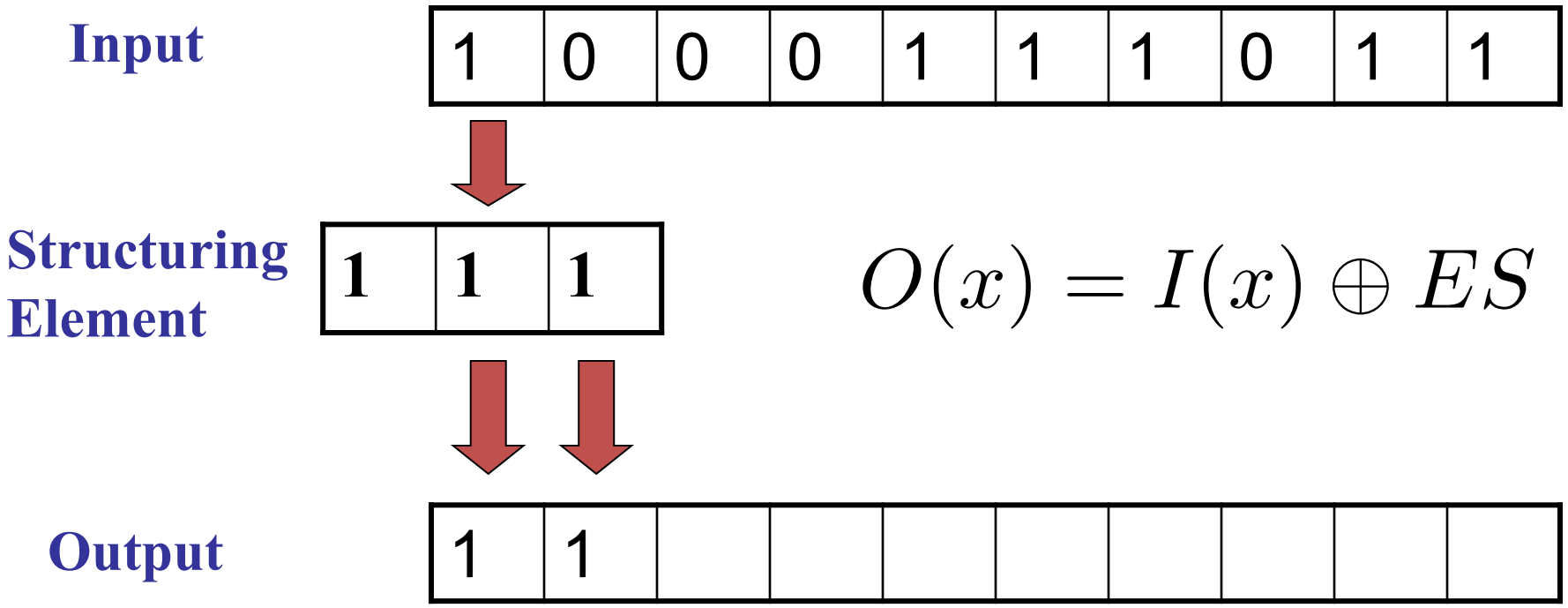
Dilation vs. Erosion

At each position:

- **Dilation**: if current pixel is foreground OR the center of structuring element covers a foreground pixel in the input image.

Slide credit: Kristen Grauman

Example for Dilation (1D)

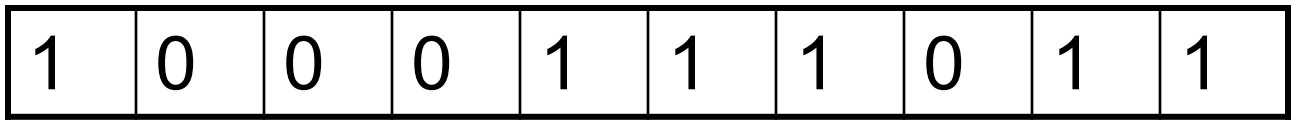


$$O(x) = I(x) \oplus ES$$

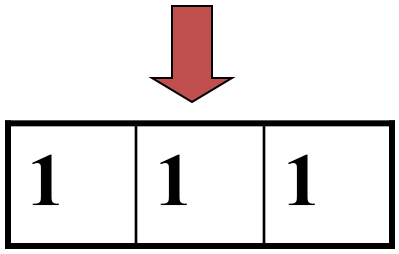
Slide credit T. Moeslund

Example for Dilation (1D)

Input



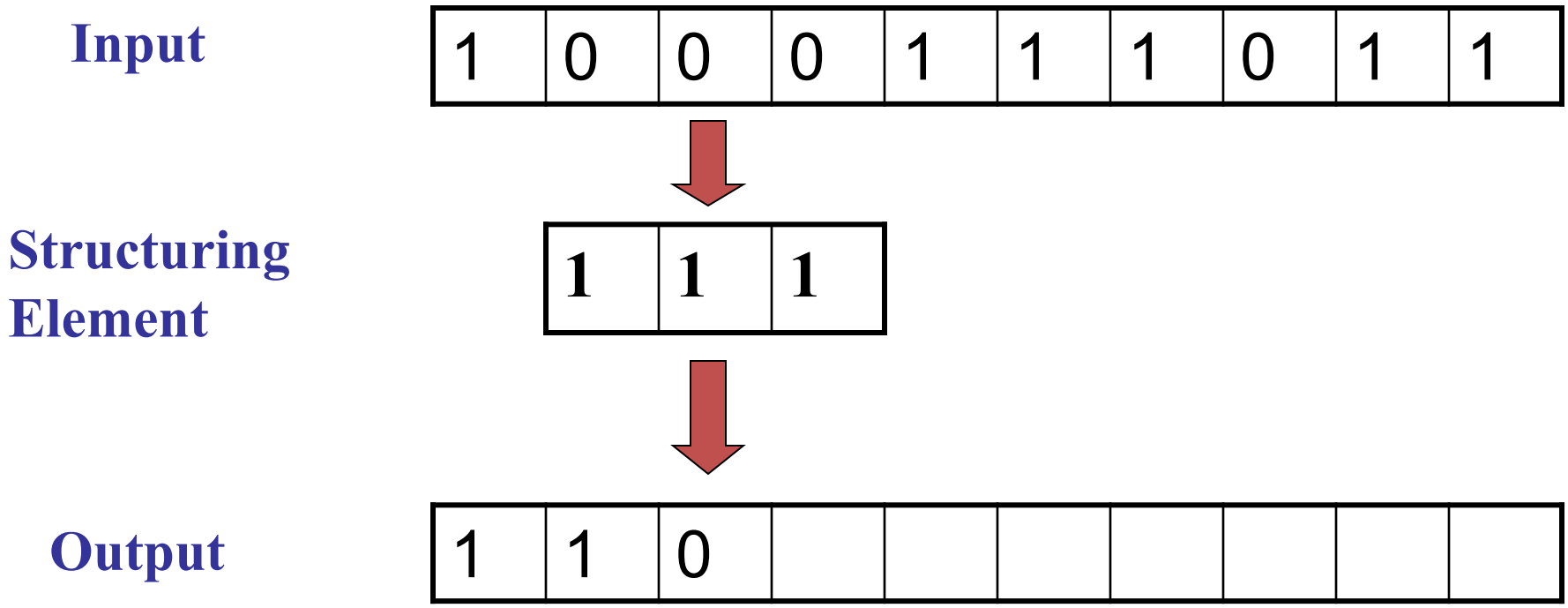
Structuring Element



Output

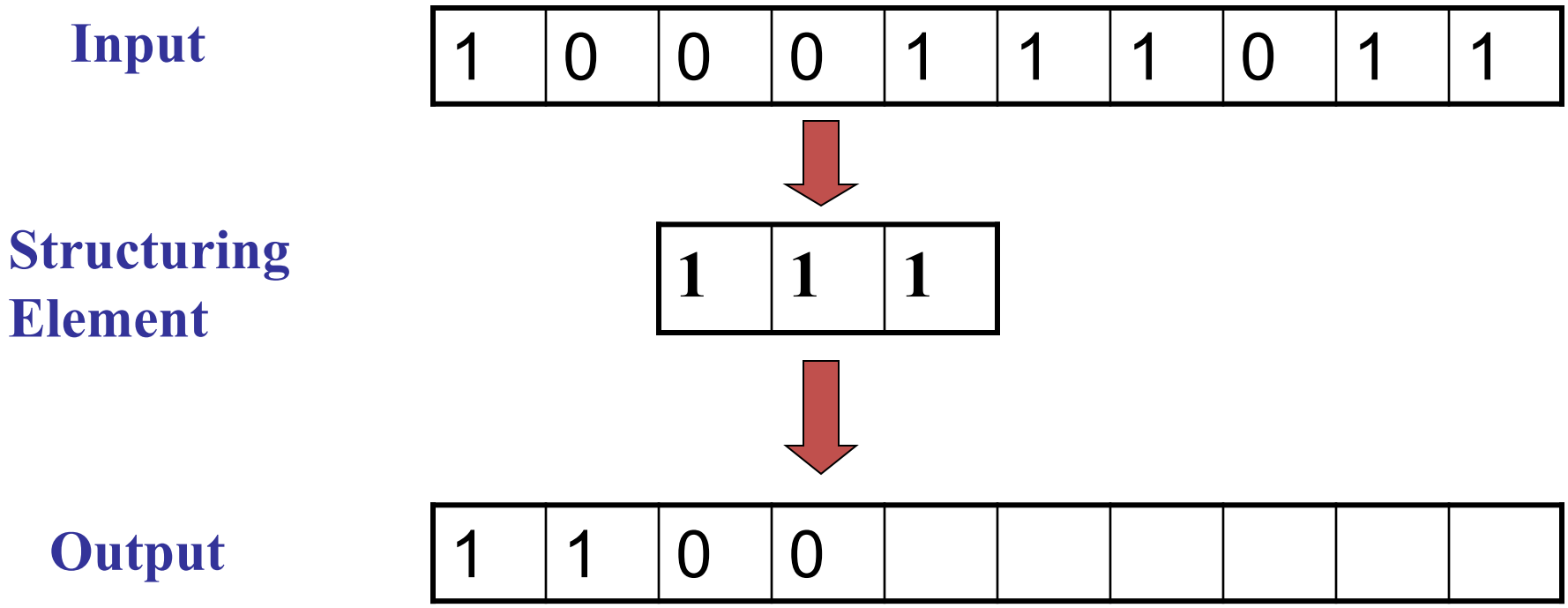
Slide credit T. Moeslund

Example for Dilation (1D)



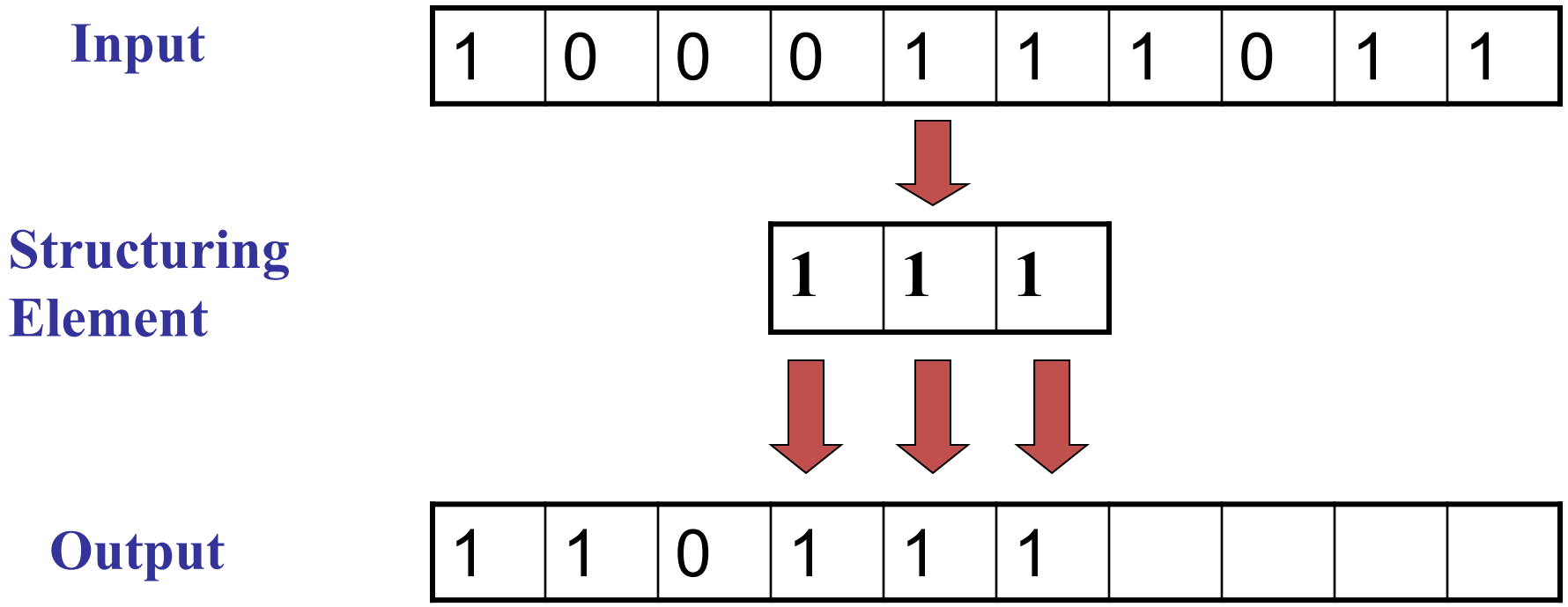
Slide credit T. Moeslund

Example for Dilation (1D)



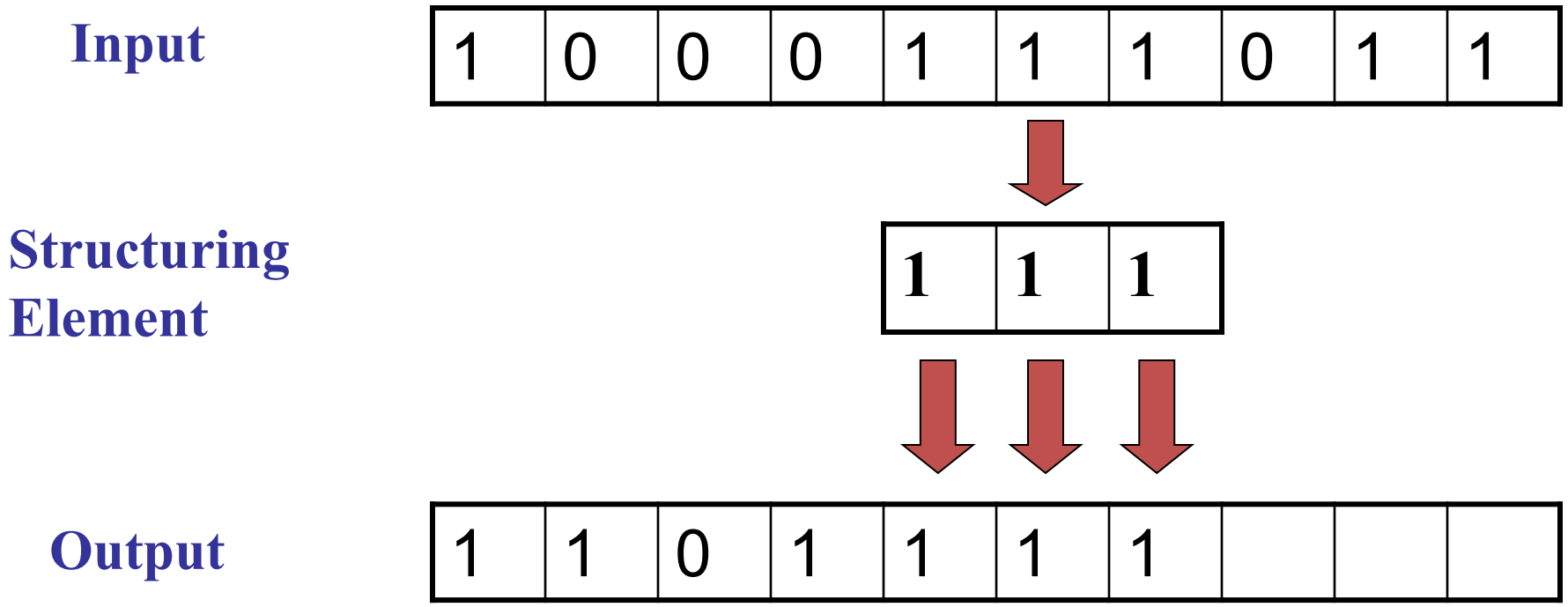
Slide credit T. Moeslund

Example for Dilation (1D)



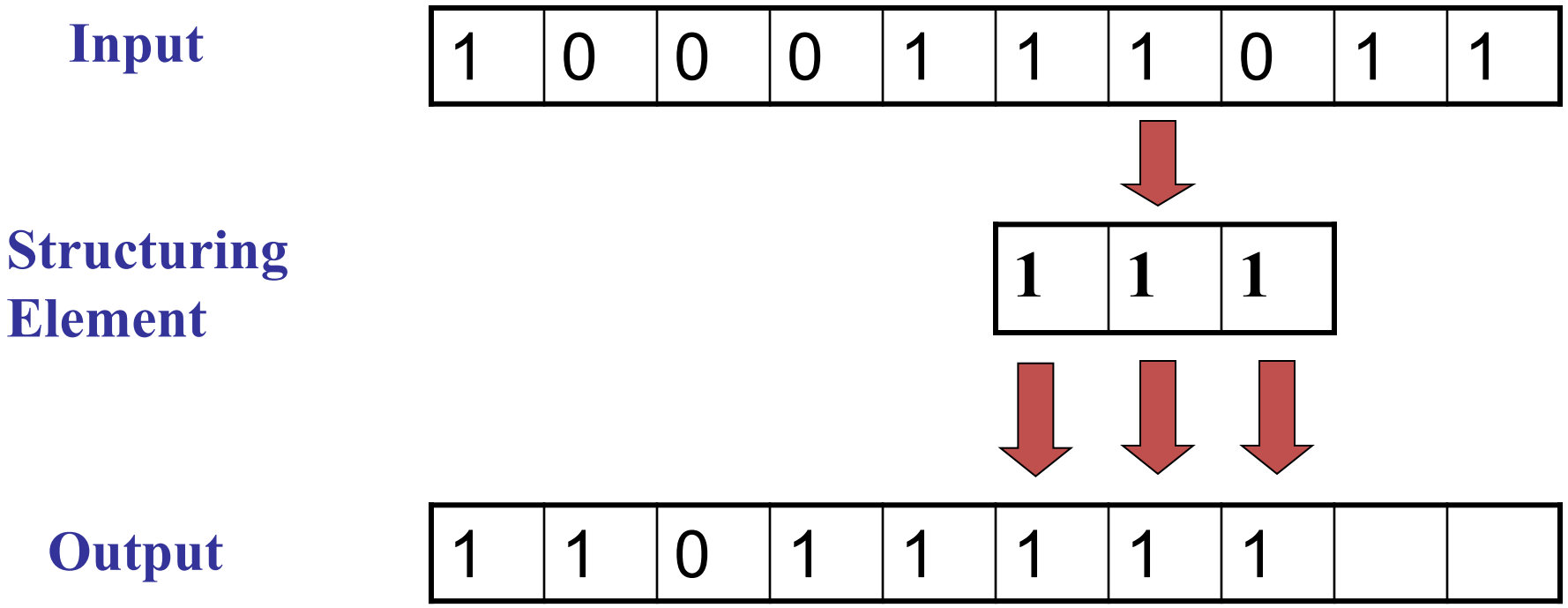
Slide credit T. Moeslund

Example for Dilation (1D)



Slide credit T. Moeslund

Example for Dilation (1D)



Slide credit T. Moeslund

Example for Dilation (1D)

Input

1	0	0	0	1	1	1	0	1	1

Structuring Element

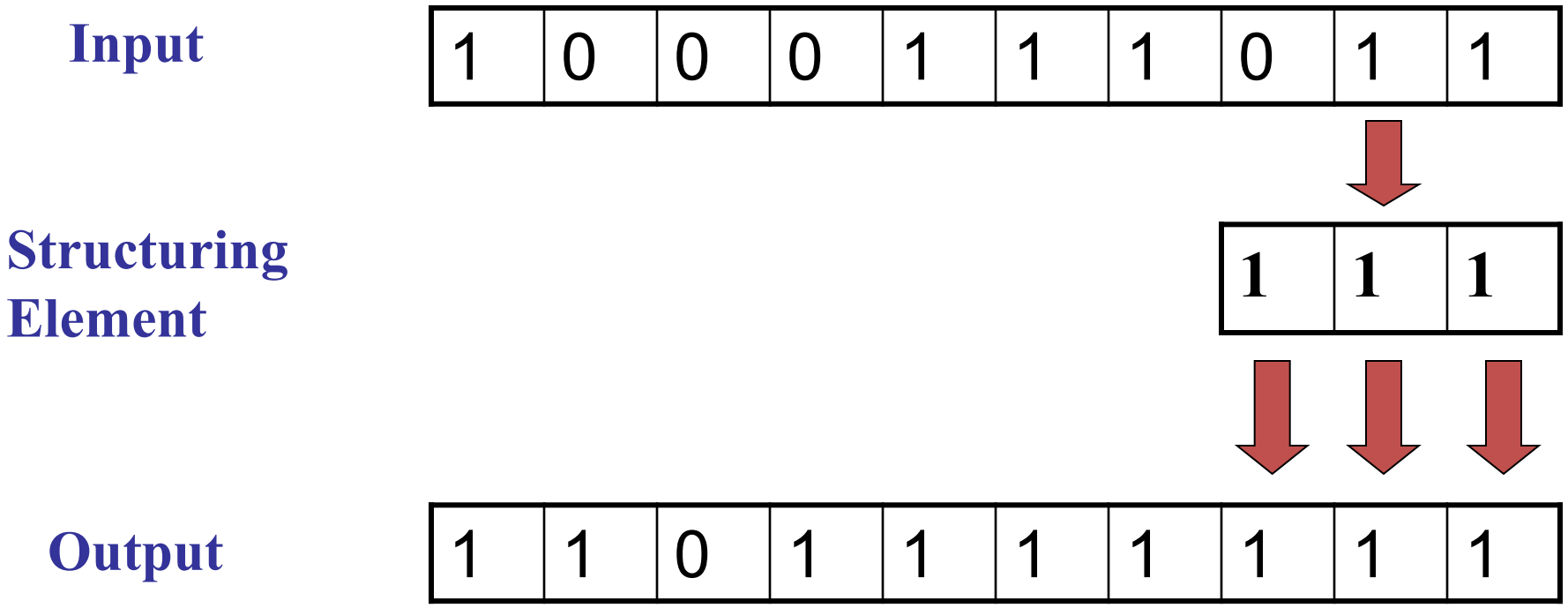
1	**1**	**1**

Output

1	1	0	1	1	1	1	1		

Slide credit T. Moeslund

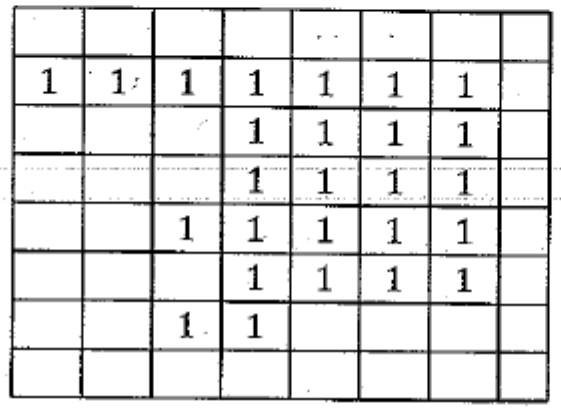
Example for Dilation (1D)



Note that the object gets bigger and holes are filled

Slide credit T. Moeslund

2D example for dilation



(*a*) Binary image **B**

(*b*) Structuring element **S**

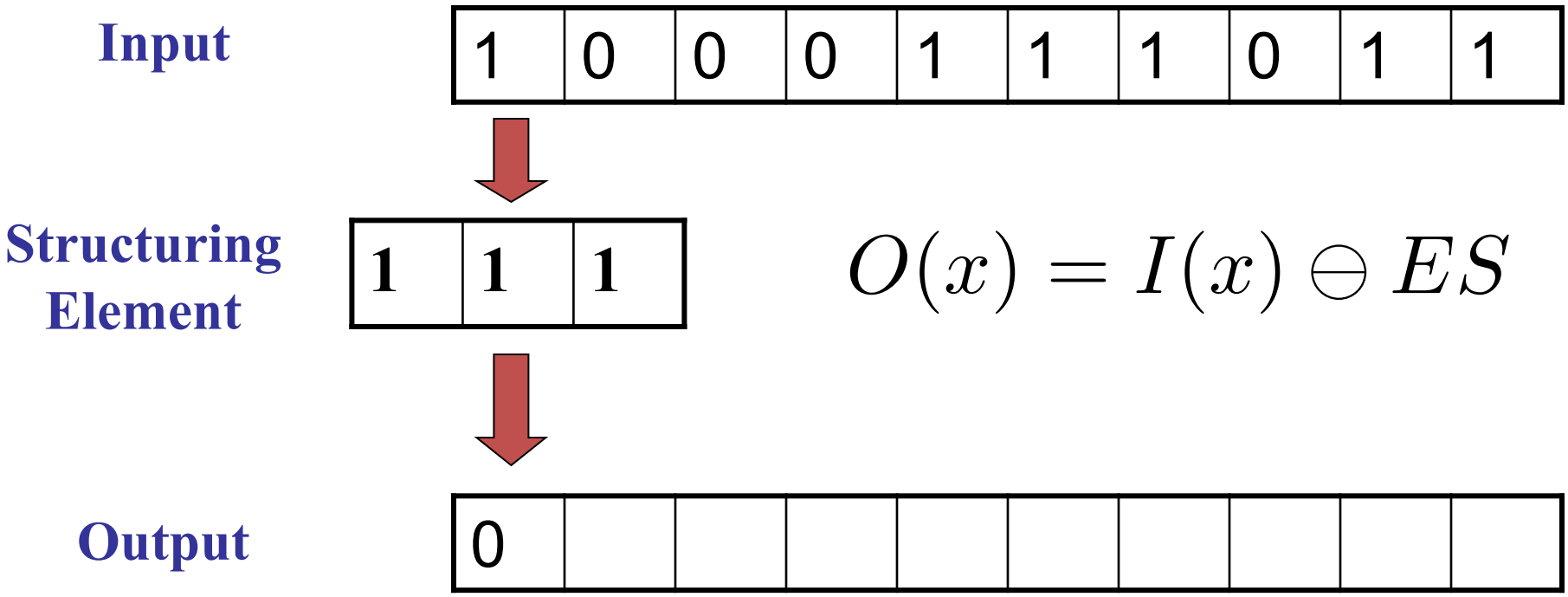
Dilation vs. Erosion

At each position:

- **Dilation**: if **current pixel** is foreground OR the center of structuring element covers foreground pixel in the input image.

- **Erosion**: if **current pixel** is foreground OR if **every pixel** under the structuring element's nonzero entries is foreground

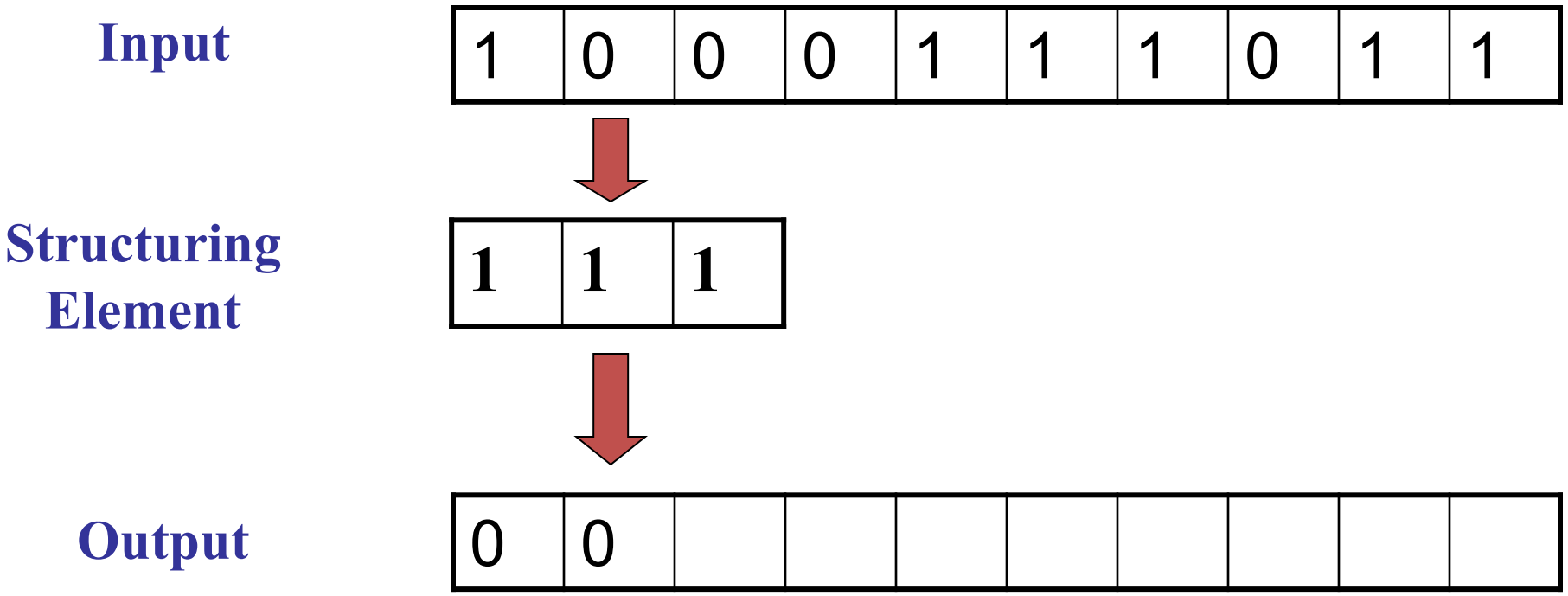
Example for Erosion (1D)



$$O(x) = I(x) \ominus ES$$

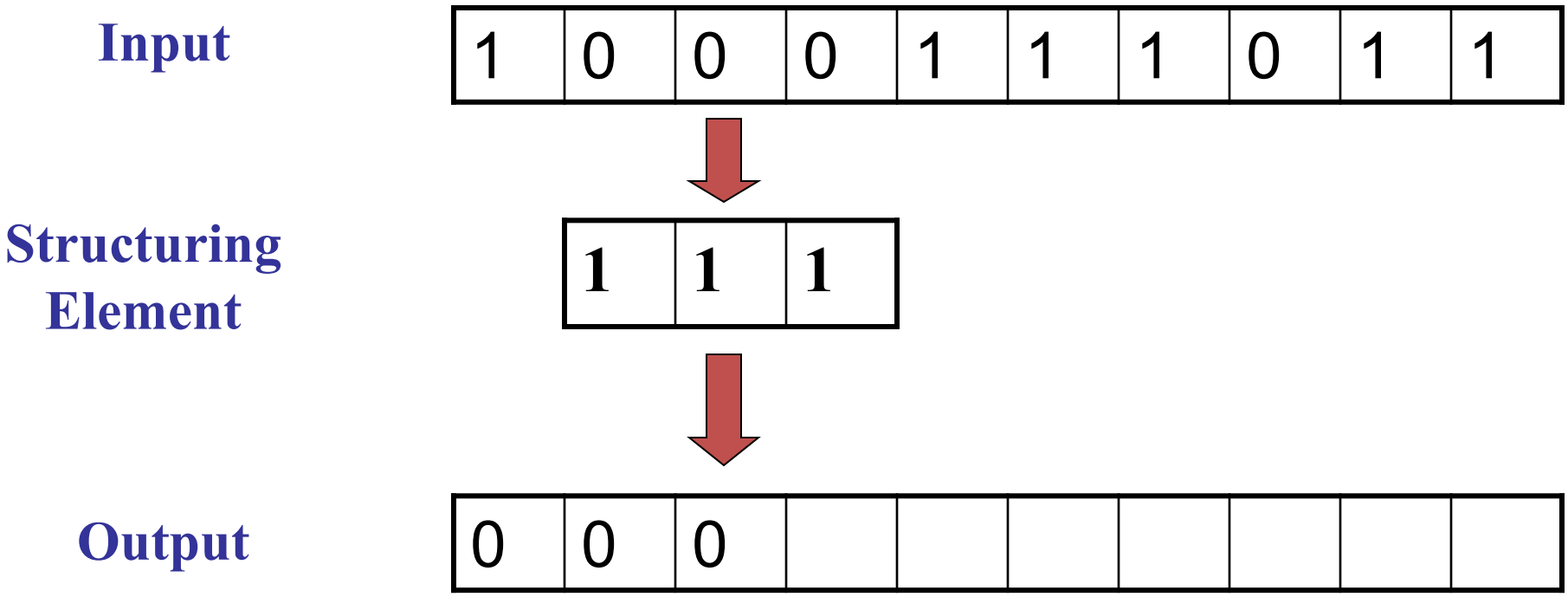
Slide credit T. Moeslund

Example for Erosion (1D)



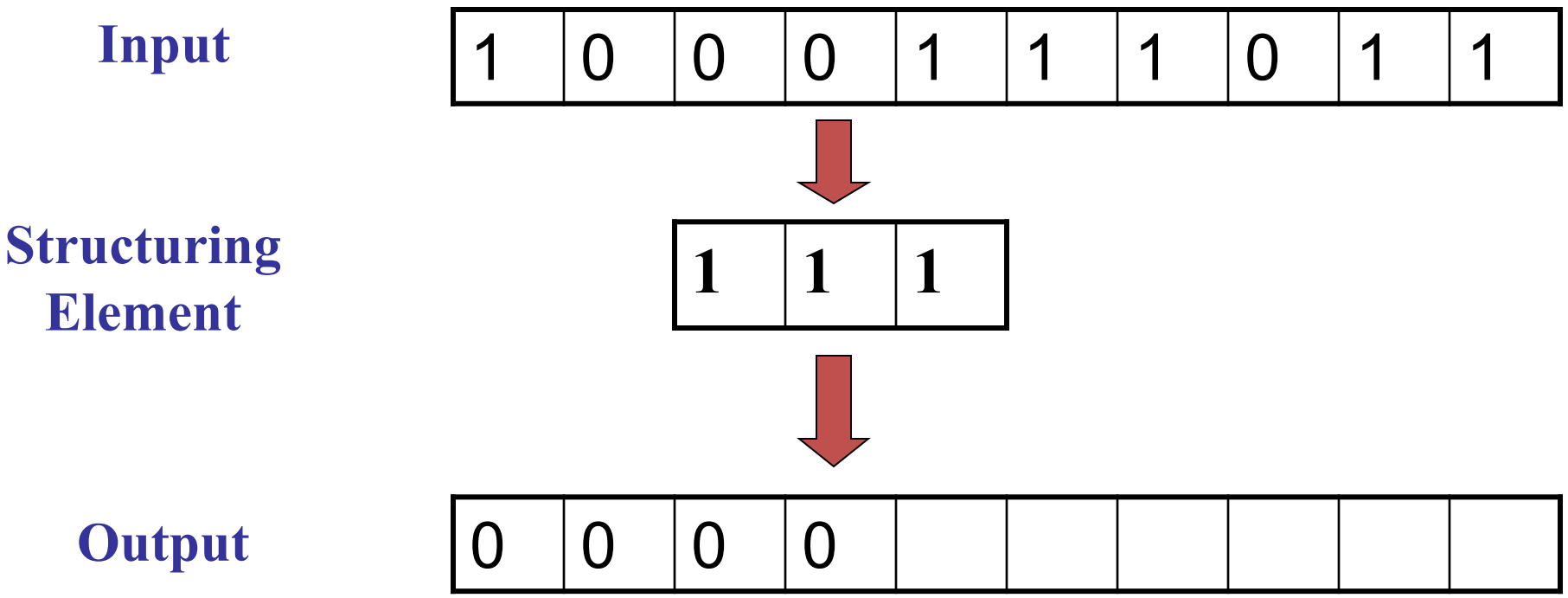
Slide credit T. Moeslund

Example for Erosion (1D)



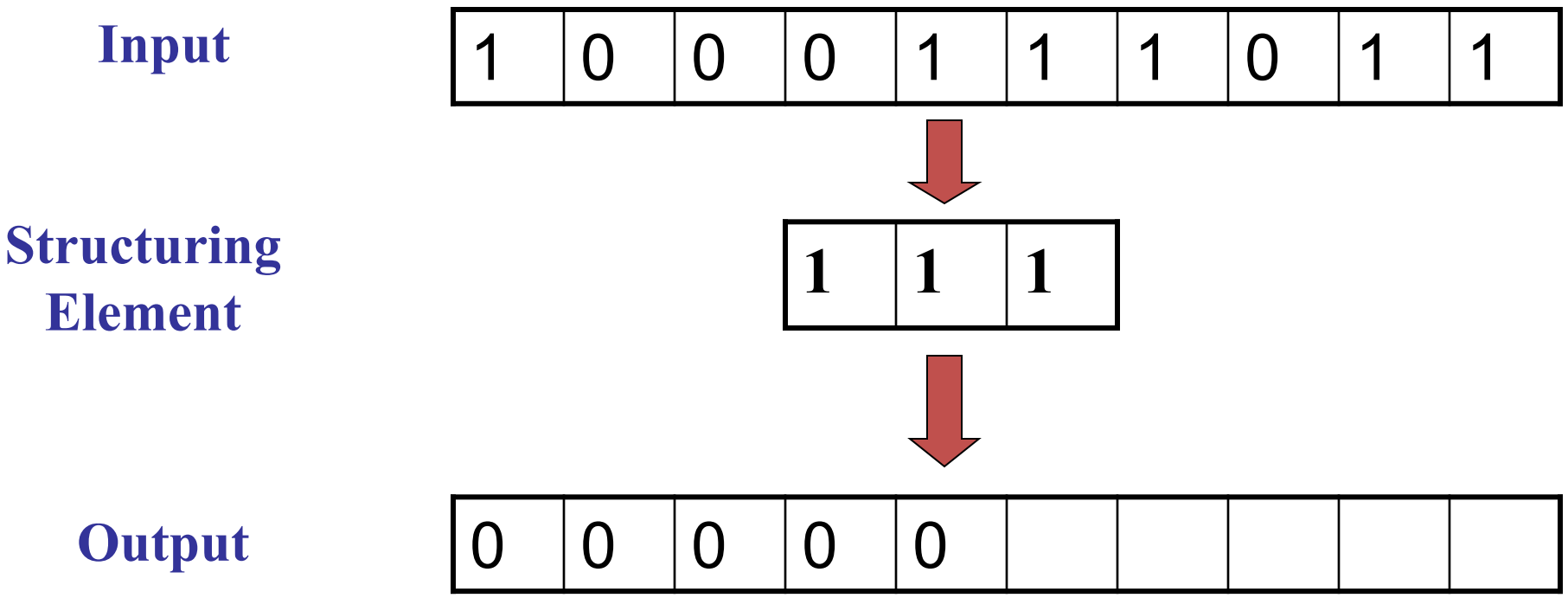
Slide credit T. Moeslund

Example for Erosion (1D)



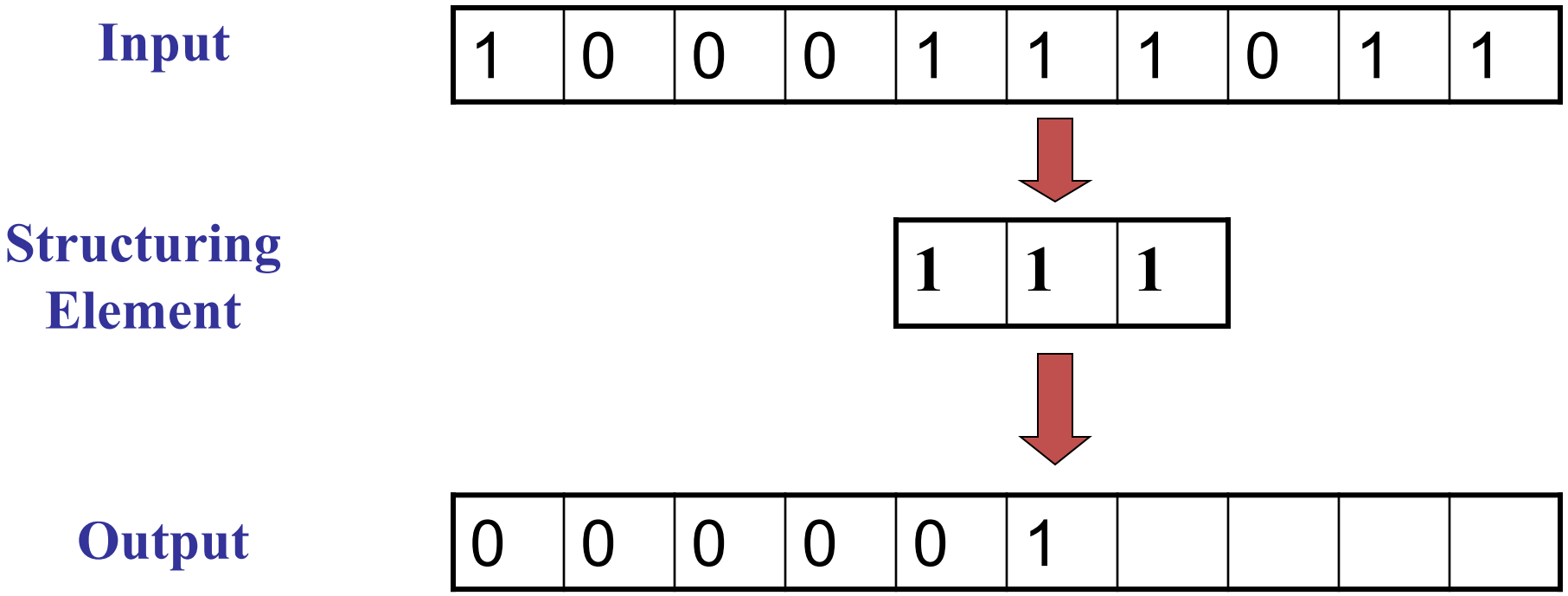
Slide credit T. Moeslund

Example for Erosion (1D)



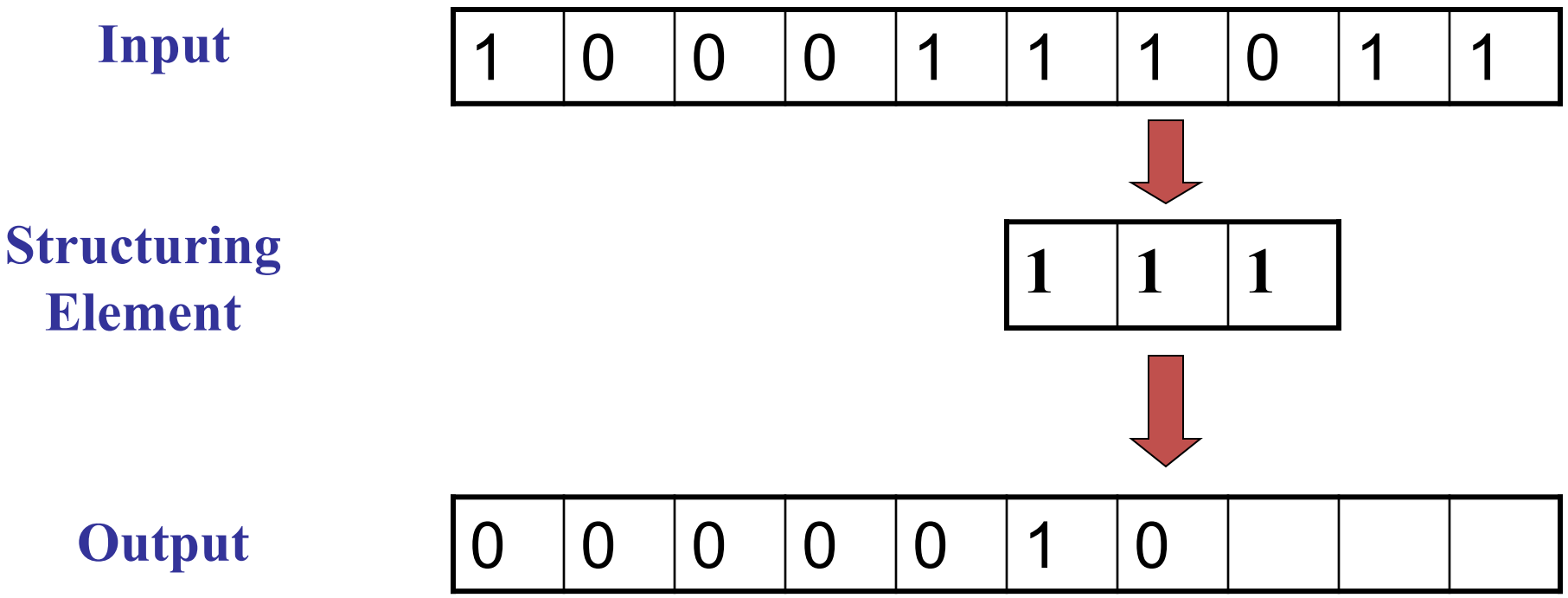
Slide credit T. Moeslund

Example for Erosion (1D)



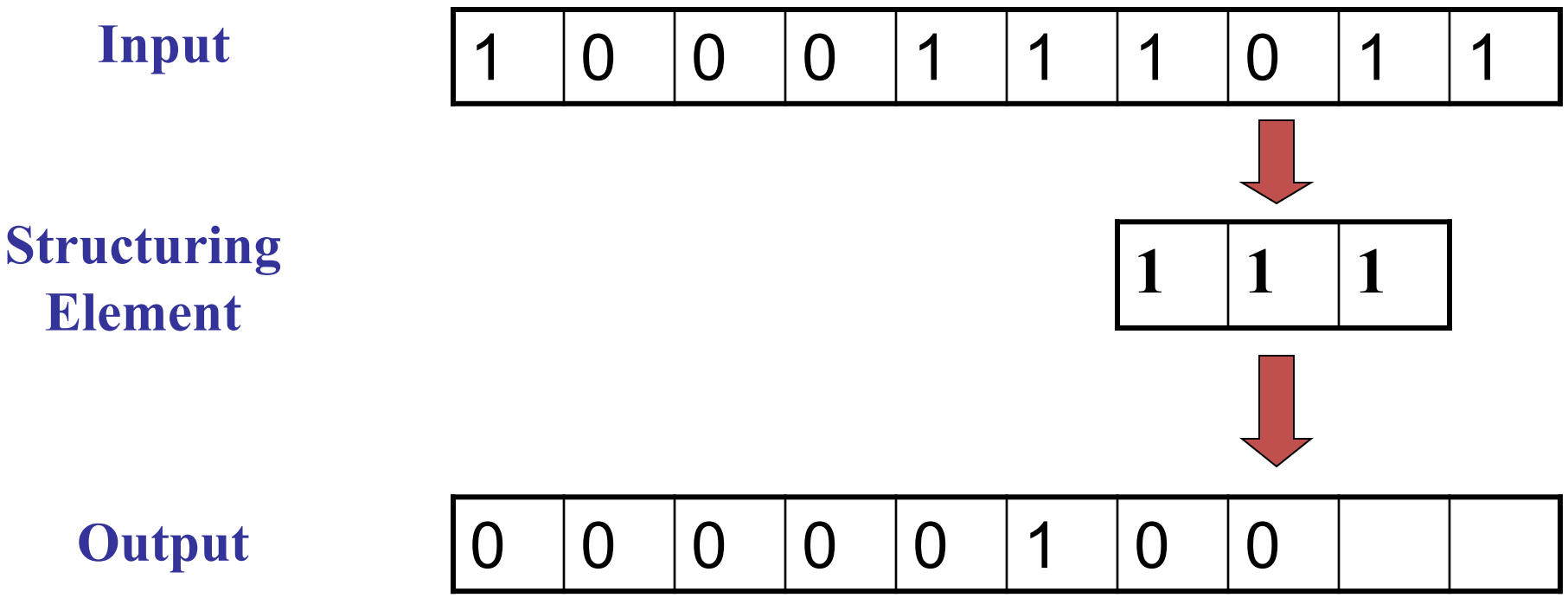
Slide credit T. Moeslund

Example for Erosion (1D)



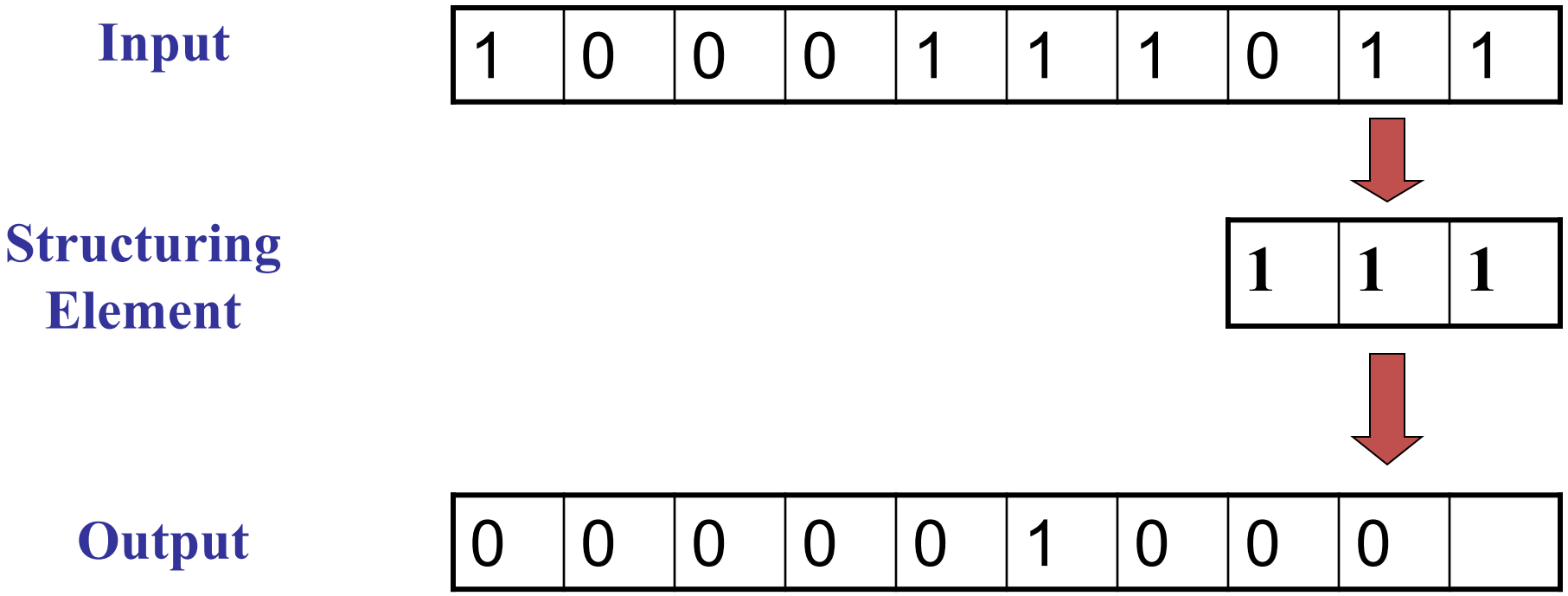
Slide credit T. Moeslund

Example for Erosion (1D)



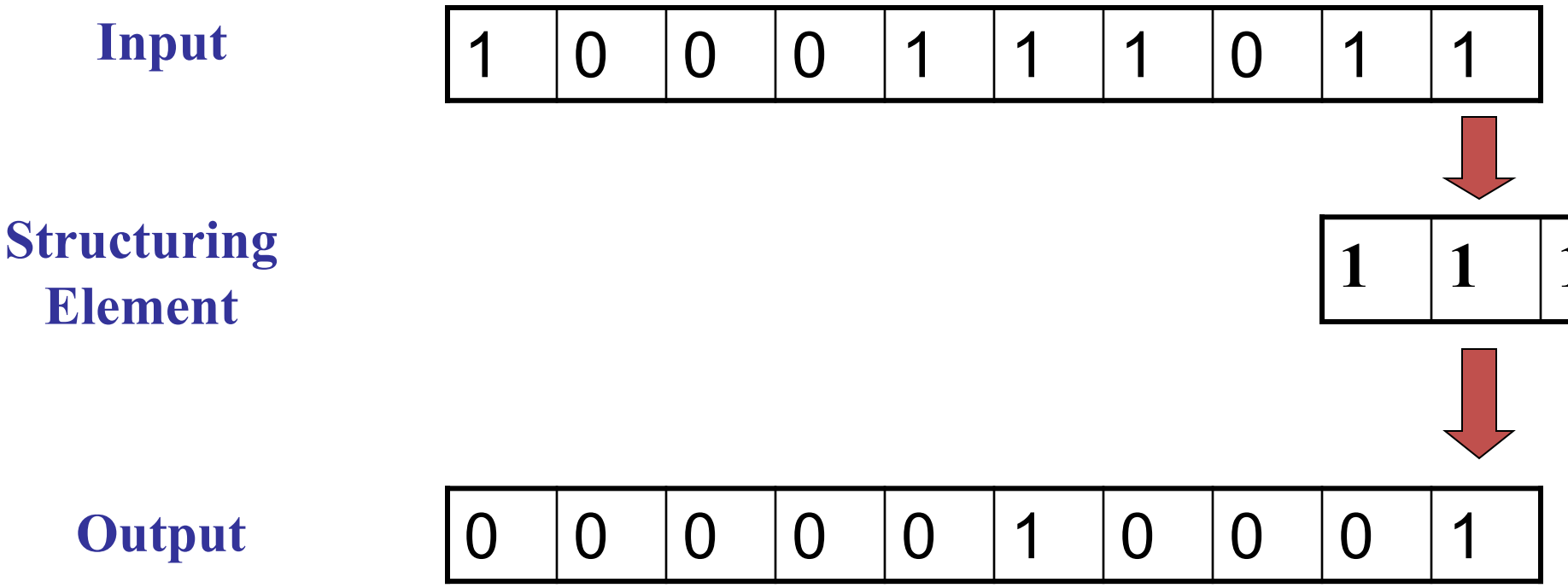
Slide credit T. Moeslund

Example for Erosion (1D)



Slide credit T. Moeslund

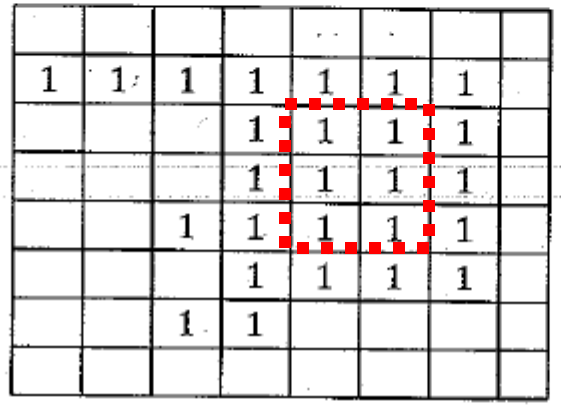
Example for Erosion (1D)



Note that the object gets smaller

Slide credit T. Moeslund

2D example for erosion

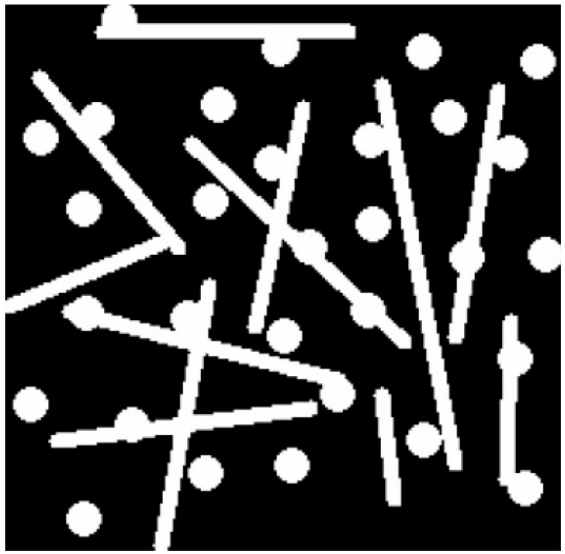


(*a*) Binary image **B**

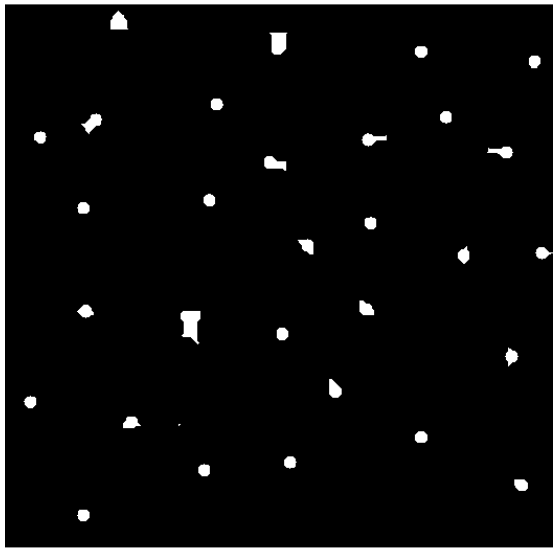
(*b*) Structuring element **S**

Opening

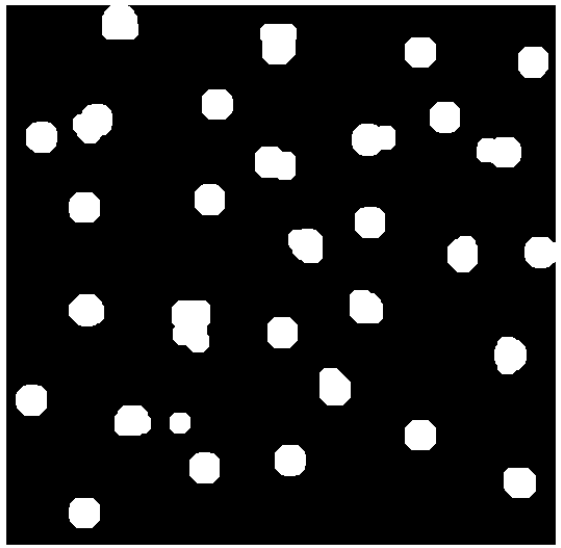
- Erode, then dilate
- Remove small objects, keep original shape



Initial image



Erosion

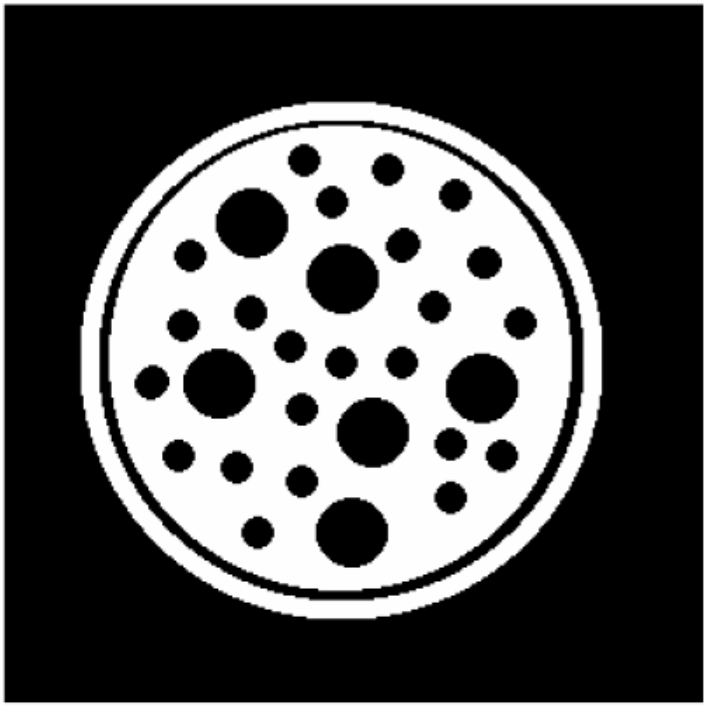


Erosion + Dilation

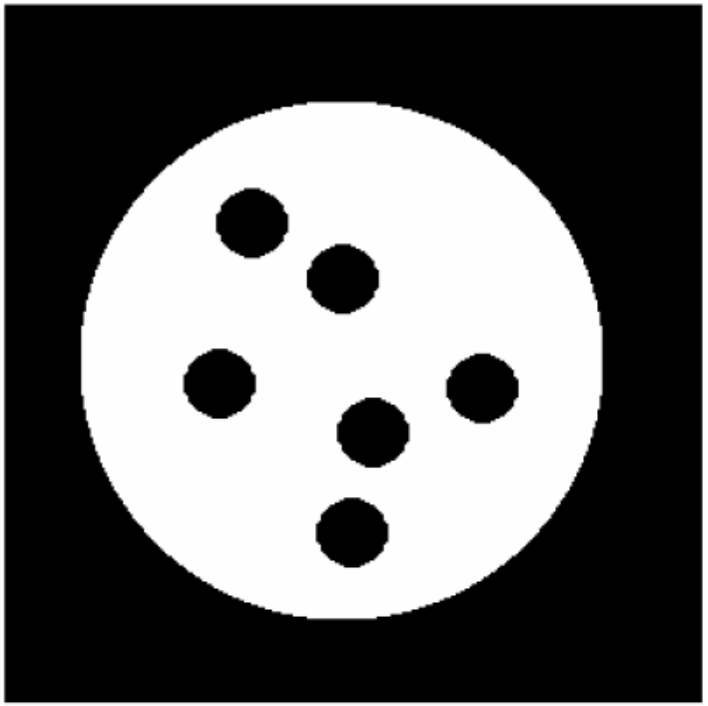
Structural element

Closing

- Dilate, then erode
- Fill holes, but keep original shape

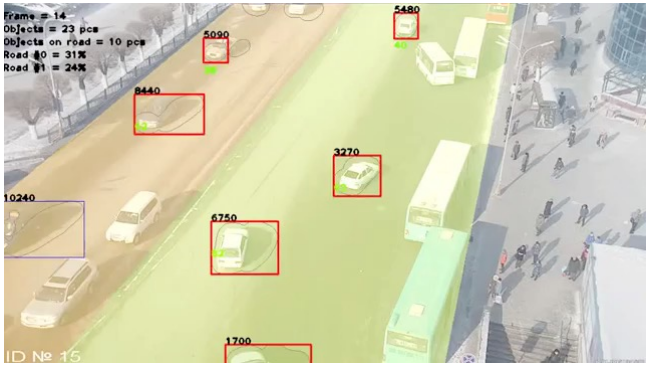


Before closing



After closing

Object tracking



traffic

body

eyes

soccer

snooker

face

Further applications

- Motion capture
- Augmented Reality
- Action Recognition
- Security, traffic monitoring
- Video Compression
- Video Summarization
- Medical Screening
- Sport Analysis

Object tracking

Optical flow is used to determine how each point is moving from frame to frame

Object tracking

Optical flow is used to determine how each point is moving from frame to frame

Different problem: track the location of target objects in each frame of a video sequence.

Object tracking

Optical flow is used to determine how each point is moving from frame to frame

Different problem: track the location of target objects in each frame of a video sequence.

We want the tracking algorithm to be insensitive to changes in:

- illumination
- scale
- rotations
- possible occlusions

Object tracking

Different problem: track the location of target objects in each frame of a video sequence.

We can build tracks for each of the object of interest and understand how objects are moving with respect to each other wrt 3D scene.

Object tracking

Different problem: track the location of target objects in each frame of a video sequence.

We can build tracks for each of the object of interest and understand how objects are moving with respect to each other wrt 3D scene.

Topics:

(1) Change Detection (detect meaningful changes in the scene)

(2) Gaussian Mixture Model

(3) Tracking by Feature Detection

(4) Object Tracking using Particle Filter

Change detection

Given: static camera observing scene (room, street, station, etc.)

Find: meaningful changes (moving objects, people, etc)

Want to build algorithms which are insensitive to uninteresting changes.

Robust and real-time classification of each pixel as foreground (motion/change) or background (static)

Change detection: challenges

Ignore uninteresting changes:

- Background fluctation
- Image noise

Change detection: challenges

Ignore uninteresting changes:

- Background fluctation
- Image noise

- Rain, snow, turbulence

Change detection: challenges

Ignore uninteresting changes:

- Background fluctation
- Image noise

- Rain, snow, turbulence

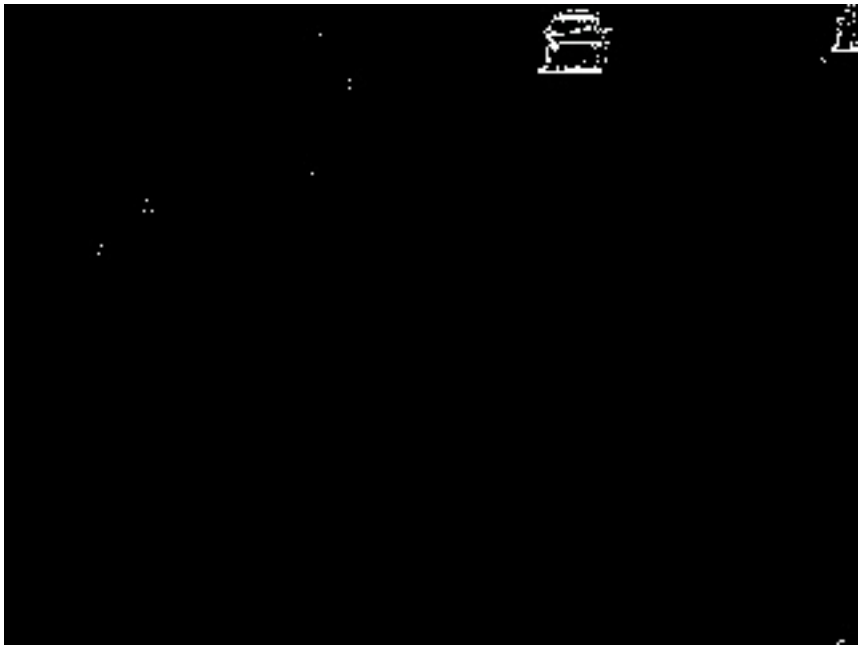
- Illumination changes & shadows
- Camera shake

Simple frame difference

Label significant difference between current and previous frame as foreground.

$$F_t = |I_t - I_{t-1}| > T, \qquad T: \text{threshold}$$

Input video sequence



Frame difference

Simple frame difference

Label significant difference between current and previous frame as foreground.

$$F_t = |I_t - I_{t-1}| > T, \qquad T: \text{threshold}$$

Input video sequence

Frame difference + dilation + erosion

Simple frame difference

Label significant difference between current and previous frame as foreground.

$$F_t = |I_t - I_{t-1}| > T, \qquad T: \text{threshold}$$

Input video sequence

Frame difference + erosion + dilation

Background modelling: Average

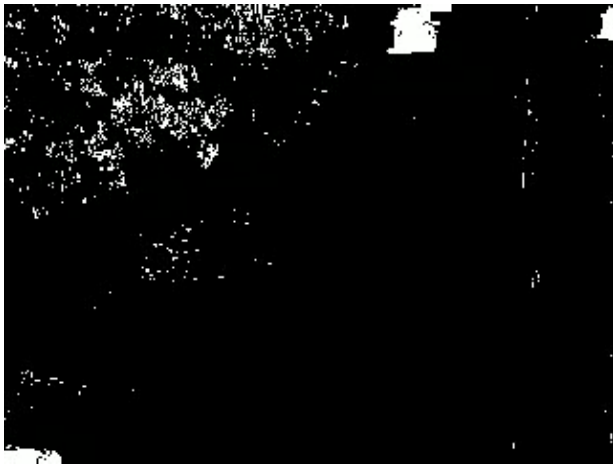
Build simple model of background before classification

Background B

average{I_1, I_2, …, I_k}

(first K frames)

Input frame I_t



Foreground F_t

$F_t = |I_t - B| > T$

Cannot handle change in background, lighting, etc.

Background modelling: Median

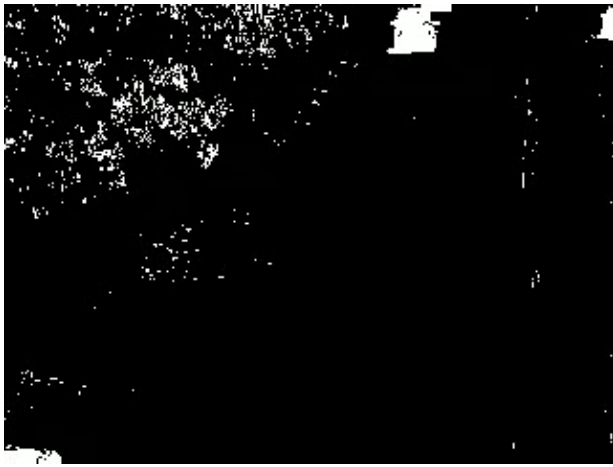
Build simple model of background before classification

Background B

median{I_1, I_2, …, I_k}

(first K frames)

Input frame I_t



Foreground F_t

$F_t = |I_t - B| > T$

Cannot handle change in background, lighting, etc.

Background modelling: Moving Median

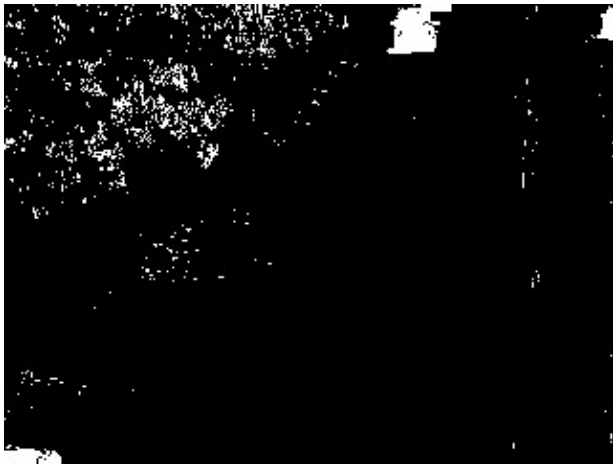
Build simple **adaptive** model of background over time.

Background B

median{I_{t-1}, I_{t-2}, …, I_{t-k}}

(last K frames)

Input frame I_t



Foreground F_t

$F_t = |I_t - B| > T$

Cannot handle significant pixel fluctuations

Mixture Model

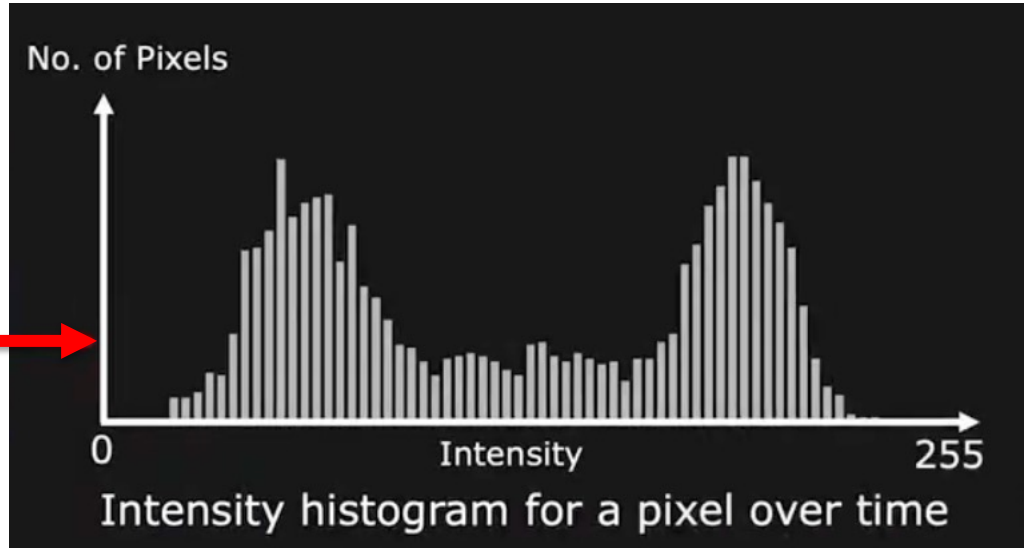
Intensity distribution at each pixel over time:

Input video sequence

Mixture Model

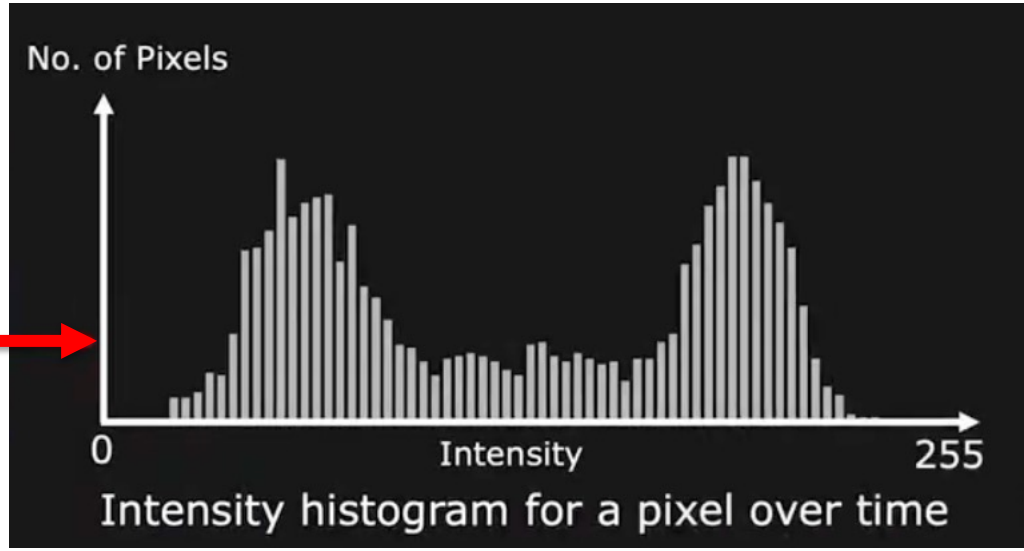
Intensity distribution at each pixel over time:

Input video sequence



Mixture Model

Intensity distribution at each pixel over time:



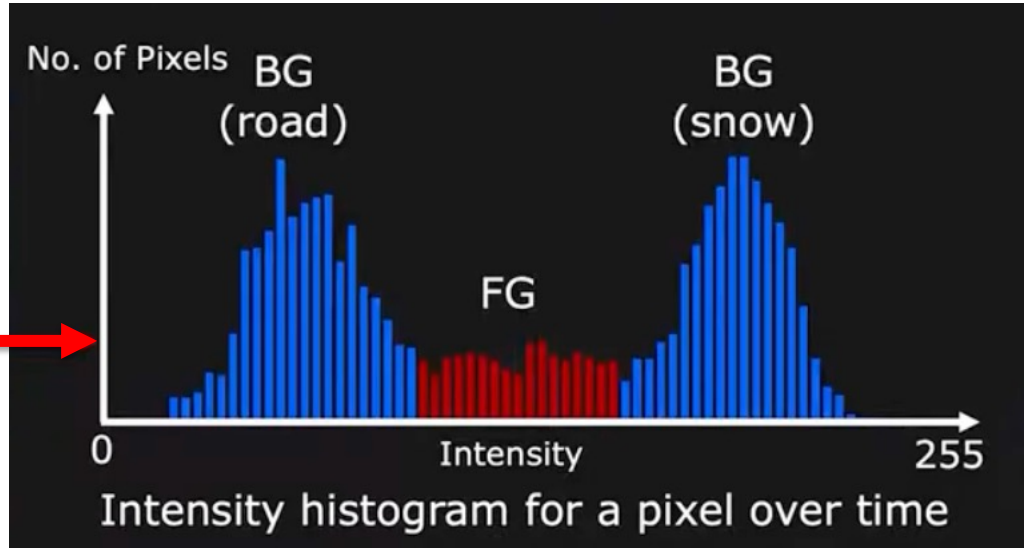
Input video sequence

Intensity variations due to static scene (road), noise (snow) and occasional moving objects (vehicles).

Mixture Model

Intensity distribution at each pixel over time:

Input video sequence

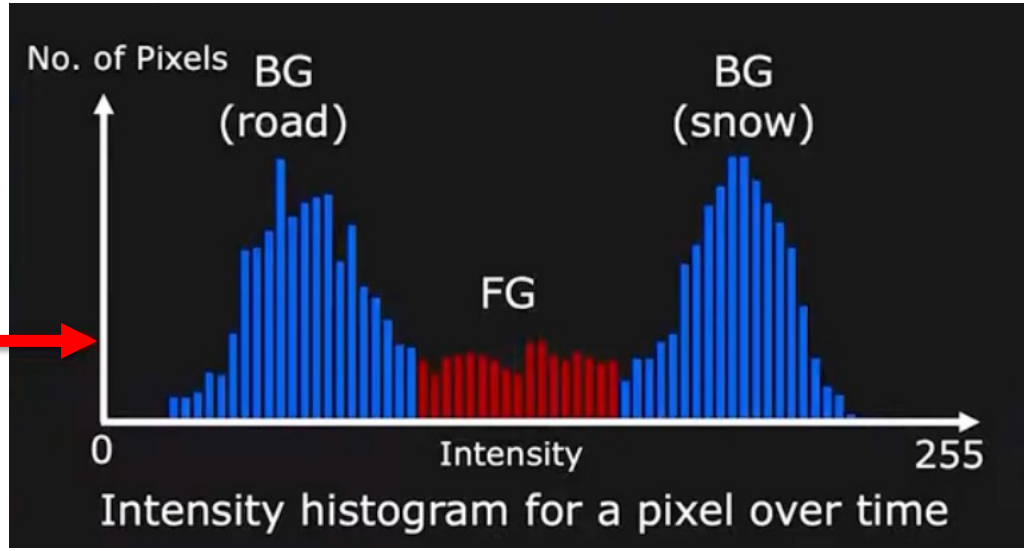


Intensity variations due to static scene (**road**), noise (**snow**) and occasional moving objects (**vehicles**).

Mixture Model

Intensity distribution at each pixel over time:

Input video sequence



Intensity variations due to static scene (**road**), noise (**snow**) and occasional moving objects (**vehicles**).

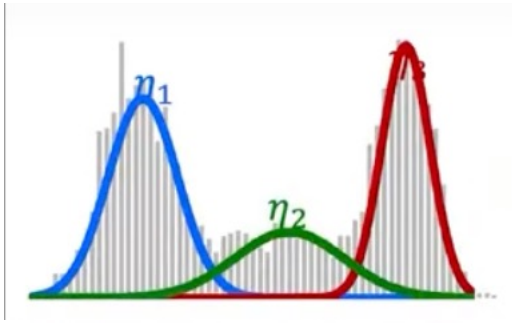
Intuition: Pixels are background most of the time

Gaussian Model

Probability Distribution
P(x) (x: pixel intensity)

Mixture of Gaussians

Probability Distribution
P(x) (x: pixel intensity)

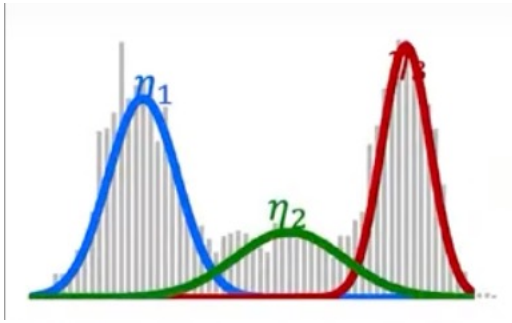


Mixture of Gaussians
$\omega_k, \eta_k(x, \mu_k, \sigma_k)$

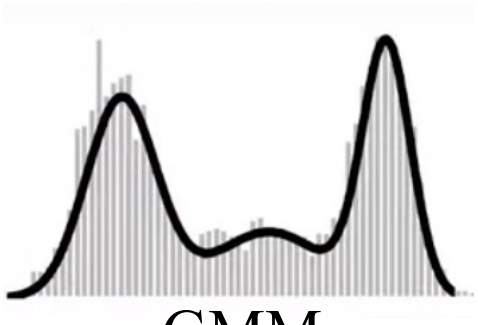
Assume P(x) is made of K different Gaussians.

Gaussian Mixture Model (GMM)

Probability Distribution
P(x) (x: pixel intensity)



Mixture of Gaussians
$\omega_k, \eta_k(x, \mu_k, \sigma_k)$



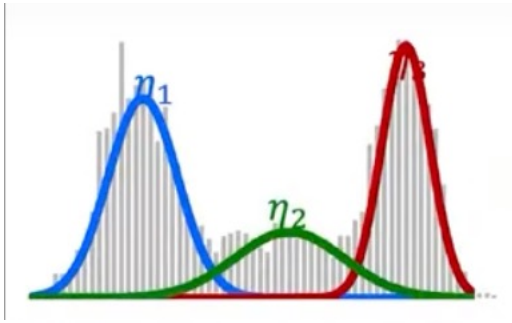
GMM
$P(x) = \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k)$

Gaussian Mixture Model Distribution : weighted sum of K Gaussians.

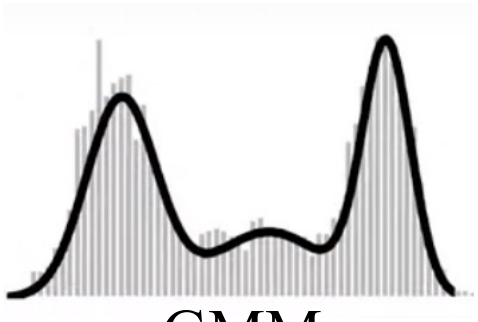
$$P(x) \cong \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k) \quad \text{such that} \quad \sum_{k=1}^{K} \omega_k = 1$$

Gaussian Mixture Model (GMM)

Probability Distribution
P(x) (x: pixel intensity)



Mixture of Gaussians
$\omega_k, \eta_k(x, \mu_k, \sigma_k)$



GMM
$P(x) = \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k)$

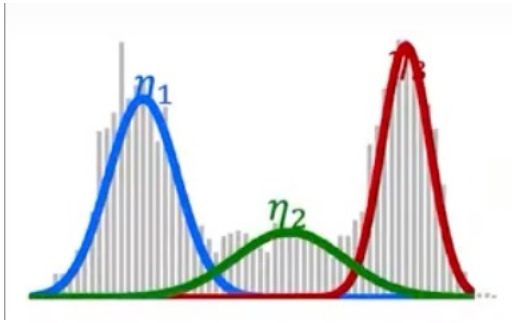
Gaussian Mixture Model Distribution : weighted sum of K Gaussians.

$$P(x) \cong \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k) \quad \text{such that} \quad \sum_{k=1}^{K} \omega_k = 1$$

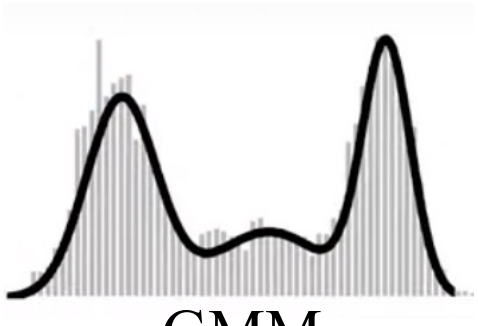
We can extend this model to high dimensional GMM, to model pixel colors (r,g,b), go from 1D to 3D

Gaussian Mixture Model (GMM)

Probability Distribution
P(x) (x: pixel intensity)



Mixture of Gaussians
$\omega_k, \eta_k(x, \mu_k, \sigma_k)$



GMM
$P(x) = \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k)$

Gaussian Mixture Model Distribution : weighted sum of K Gaussians.

$$P(x) \cong \sum_{k=1}^{K} \omega_k \eta_k(x, \mu_k, \sigma_k) \quad \text{such that} \quad \sum_{k=1}^{K} \omega_k = 1$$

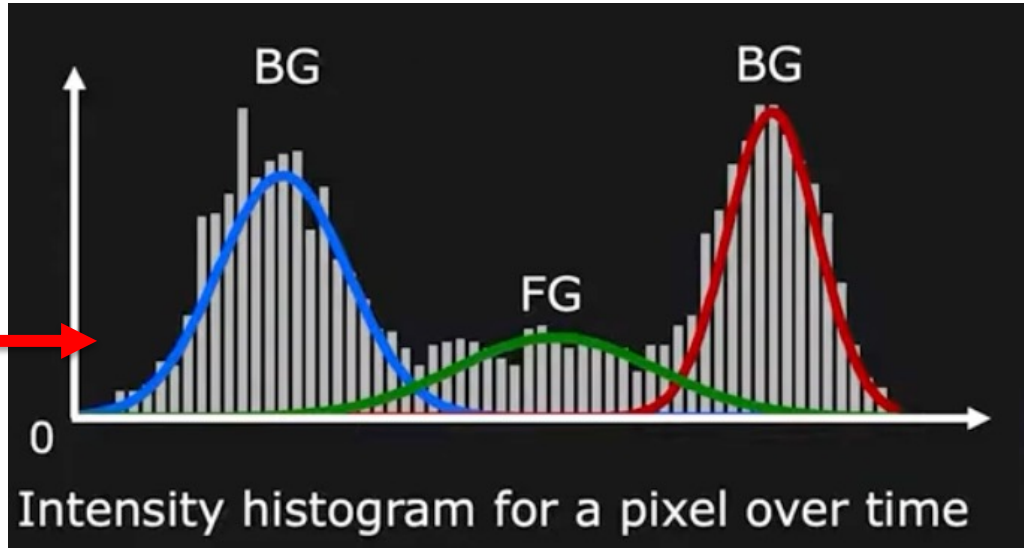
GMM can be estimated from P(x)

Background Modeling with GMM

Given: A GMM for intensity/color variation at a pixel over time

Classify: Individual Gaussians as foreground/background

Input video sequence



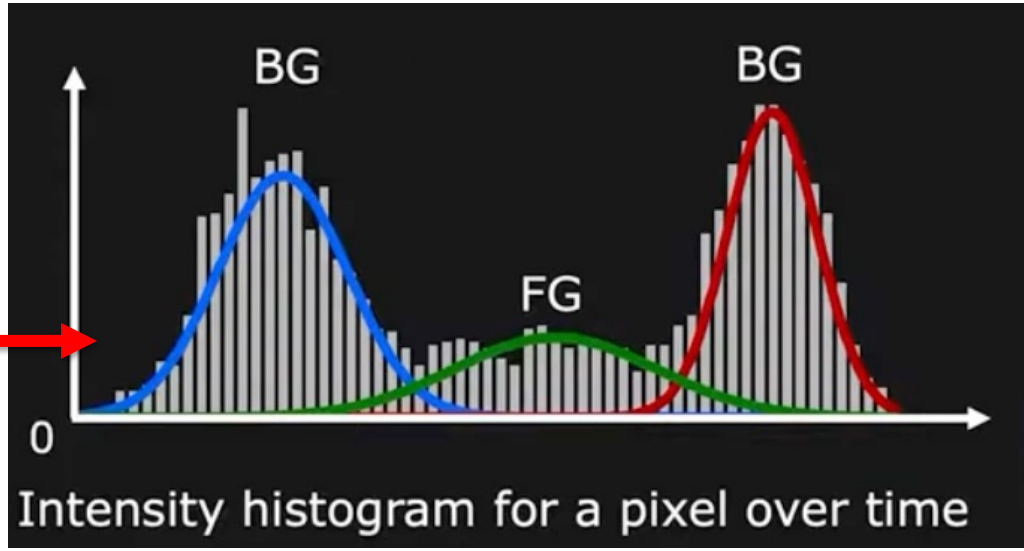
Intuition: Pixels are background most of the time. That is, Gaussians with large supporting evidence ω and small σ.

Background Modeling with GMM

Given: A GMM for intensity/color variation at a pixel over time

Classify: Individual Gaussians as foreground/background

Input video sequence



Intuition: Pixels are background most of the time. That is, Gaussians with large supporting evidence ω and small σ.

Large $\frac{\omega}{\sigma}$: Background **Small $\frac{\omega}{\sigma}$: Foreground**

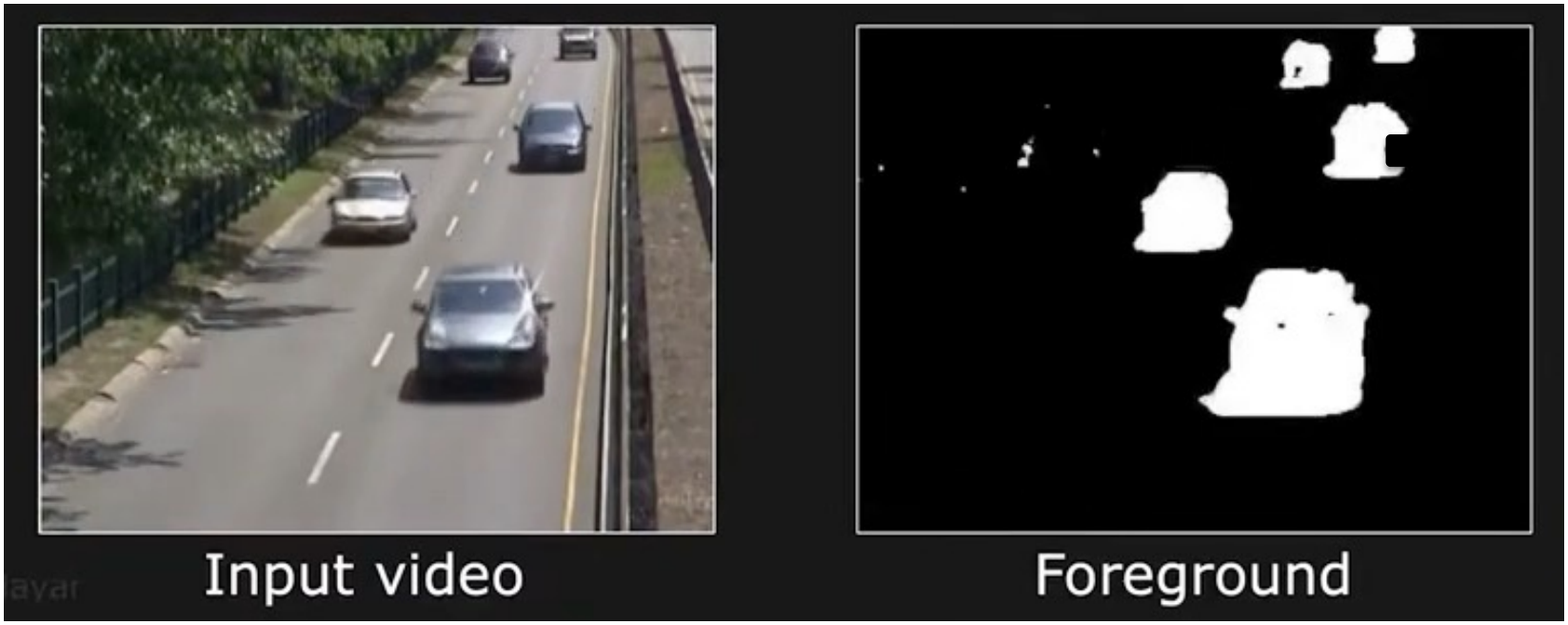
Change Detection using GMM

For each pixel:

1. Compute pixel color histogram H using first N frames.
2. Normalize histogram H: $\overline{H} = \frac{H}{||H||}$
3. Model $\overline{H}$ as a mixture of K (3 to 5) Gaussians.
4. For each susequent frame:
    1. The pixel value X belong to Gaussian k in GMM for which $\|X - \mu_k\|$ is minimum but also $\|X - \mu_k\| < 2.5\sigma_k$
    2. If $\frac{\omega_k}{\sigma_k}$ is large the classify pixel as background. Else classify as foreground.
    3. Update histogram H using new pixel intensity
    4. If $\overline{H}$ and $\frac{H}{||H||}$ differ a lot the refit GMM

Adaptive Background Mixture Models for Real-Time Tracking, Chris Stauffer & W.E.L. Grimson, CVPR 1998

Adaptive GMM based Change Detection using GMM



Strategies for tracking

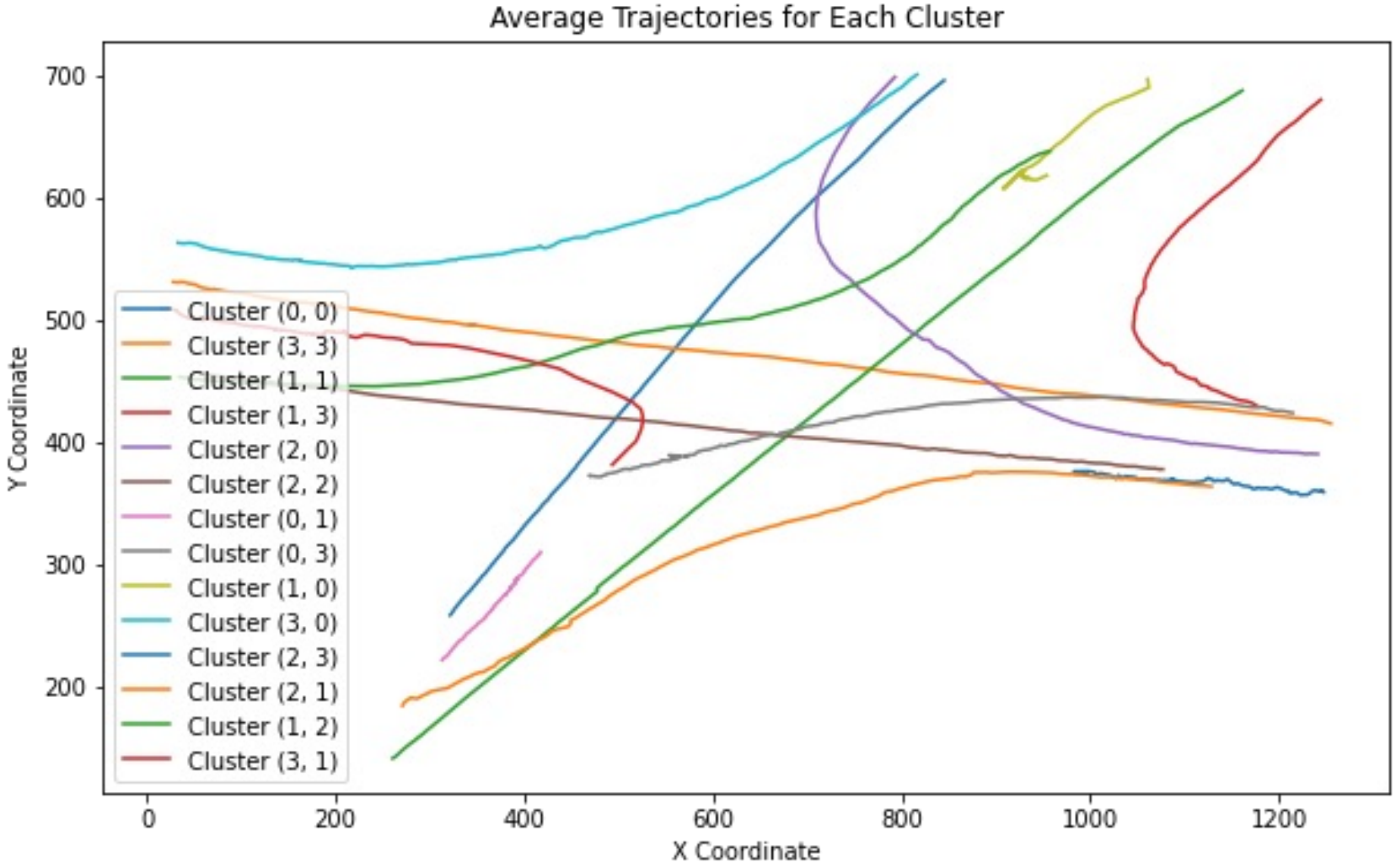
- **Feature-based tracking**
  - extract visual features (corners, textured areas) and track them over multiple frames
  - sparse motion fields, but more robust tracking
  - figure out which features can be tracked, background vs foreground model
  - some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)

- **Tracking by repeated detection**
  - works well if object is easily detectable (e.g., face, ball or colored glove) and there is only one
  - need some way to link up detections (obtain tracks)
  - best you can do, if you can't predict motion

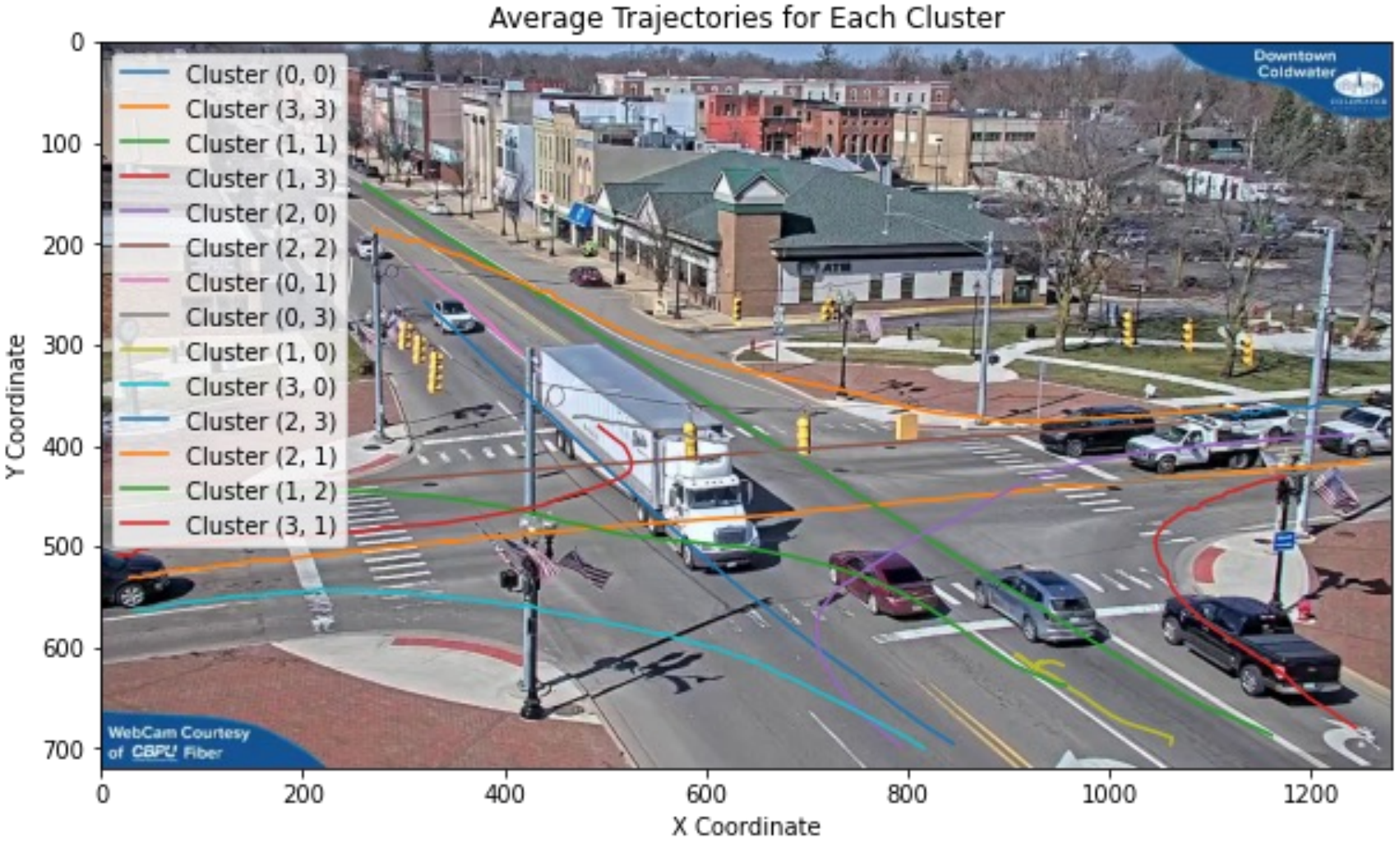
- **Tracking with dynamics**
  - based on a model of expected motion, predict where objects will occur in next frame, before even seeing the image
  - restrict search for the object
  - measurement noise is reduced by trajectory smoothness

Lab 7 – Extracting trajectorie in traffic surveillence

Lab 7 – Extracting trajectorie in traffic surveillence

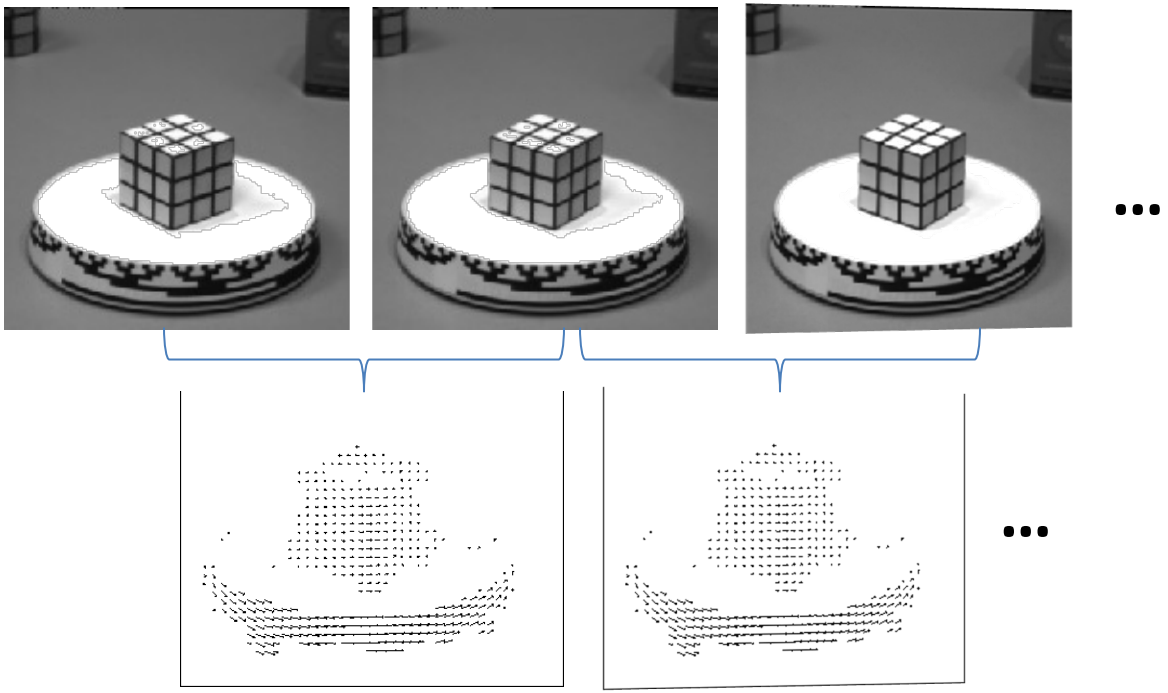


Lab 7 – Extracting trajectorie in traffic surveillence



Optical flow for tracking?

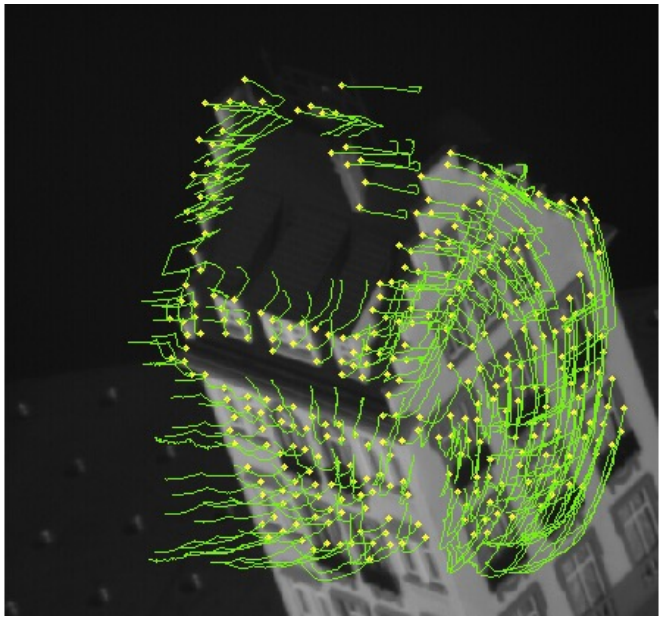
If we have more than just a pair of frames, we could compute flow from one to the next:



But flow only reliable for small motions, and we may have occlusions, textureless regions that yield bad estimates anyway…

Feature tracking

- If we have more than two images, we can track a feature from one frame to the next by following the optical flow
- Challenges
  - Finding good features to track
  - Adding and deleting tracks (trajectories of objects in video)

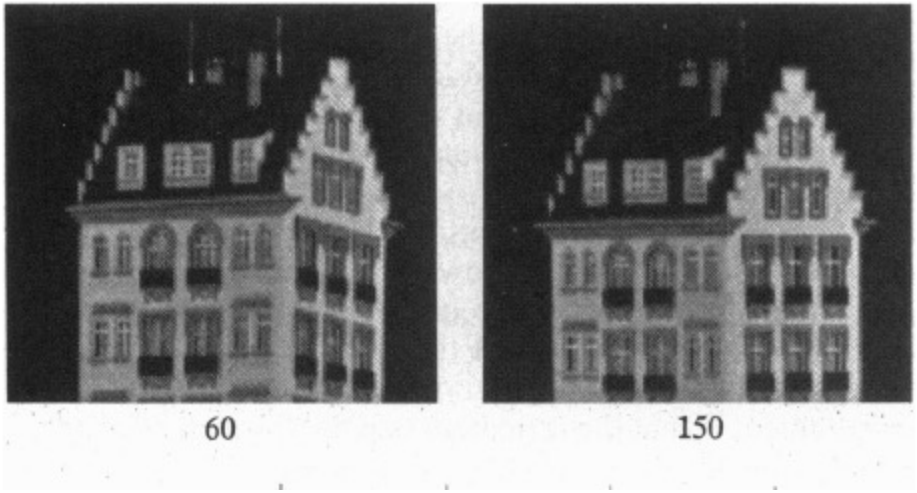


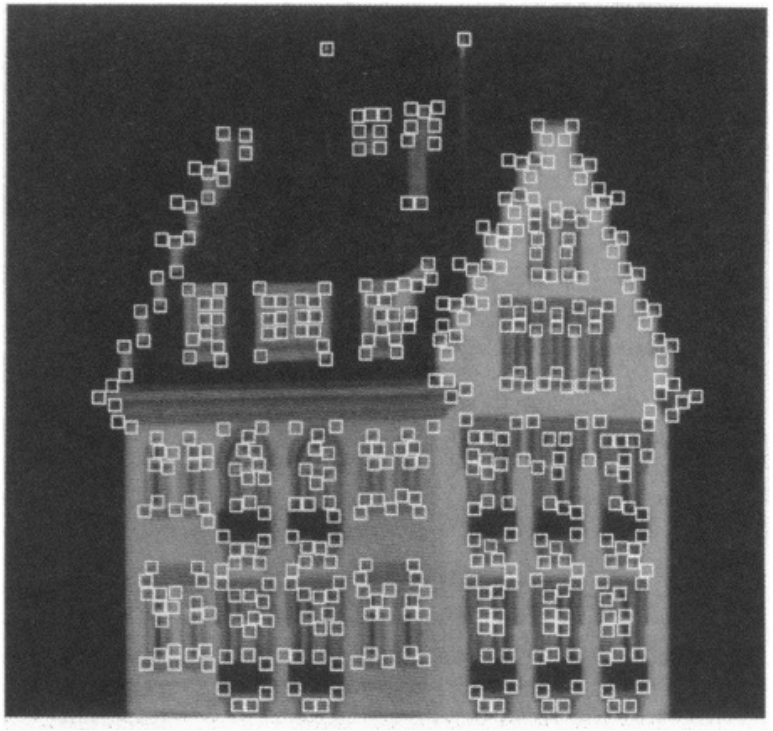
Feature tracking

- Challenges
  - Figure out which features can be tracked
  - Efficiently track across frames
  - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  - Drift: small errors can accumulate as appearance model is updated
  - Points may appear or disappear: need to be able to add/delete tracked points

Lucas-Kanade feature tracker

- Same as Lucas-Kanade optical flow, but instead for all pixels just for selected features (e.g. Harris corners)





Shi-Tomasi feature tracker

- Find good features using eigenvalues of second-moment matrix
  - Key idea: "good" features to track are the ones whose motion can be estimated reliably (Harris corner)
- From frame to frame, track with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement
- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Tracking example

Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Detection vs. tracking


t=1


t=2

...


t=20


t=21

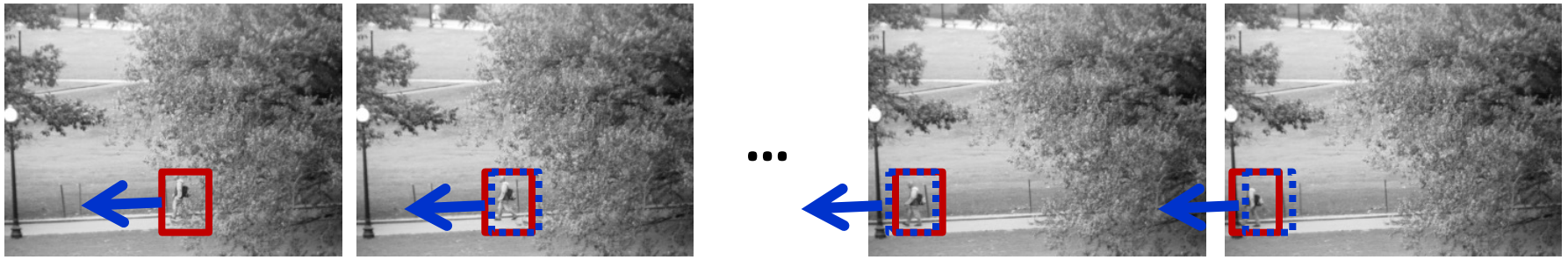
Slide credit: Kristen Grauman

Detection vs. tracking

Detection: We detect the object independently in each frame and can record its position over time, e.g., based on blob's centroid or detection window coordinates

Slide credit: Kristen Grauman

Detection vs. tracking



Tracking with *dynamics*: We use image measurements to estimate position of object, but also incorporate position predicted by dynamics, i.e., our expectation of object's motion pattern.

Slide credit: Kristen Grauman

Tracking with dynamics

- Use model of expected motion to *predict* where objects will occur in next frame, even before seeing the image.
- **Intent**:
  - Do less work looking for the object, restrict the search.
  - Get improved estimates since measurement noise is tempered by smoothness, dynamics priors.
- **Assumption**: continuous motion patterns:
  - Camera is not moving instantly to new viewpoint
  - Objects do not disappear and reappear in different places in the scene
  - Gradual change in pose between camera and scene

Slide credit: Kristen Grauman

Tracking as inference

- The *hidden state* consists of the true parameters we care about, denoted *X (*e.g. a vector containing the position, velocity, acceleration).

- The *measurement* is our noisy observation that results from the underlying state, denoted *Y*. Could be the position of some image points, the position of some image regions or pretty much anything else.

- At each time step, state changes (from X_{t-1} to X_t) and we get a new observation Y_t.

Slide credit: Kristen Grauman

Tracking as inference

- The *hidden state* consists of the true parameters we care about, denoted *X*.
- The *measurement* is our noisy observation that results from the underlying state, denoted *Y*.
- At each time step, state changes (from X_{t-1} to X_t) and we get a new observation Y_t.
- Our goal: recover most likely state X_t given
  - All observations seen so far.
  - Knowledge about dynamics of state transitions.

Slide credit: Kristen Grauman

Tracking as inference: intuition

Time t Time t+1

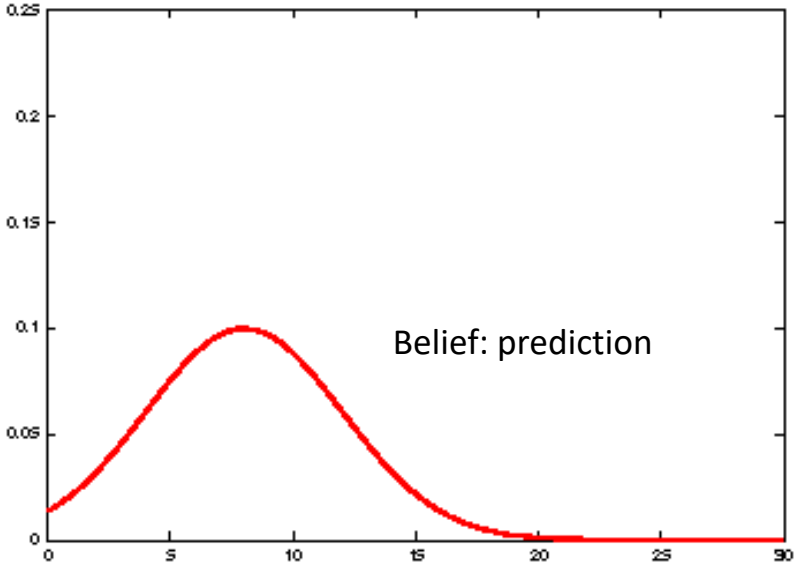
Belief

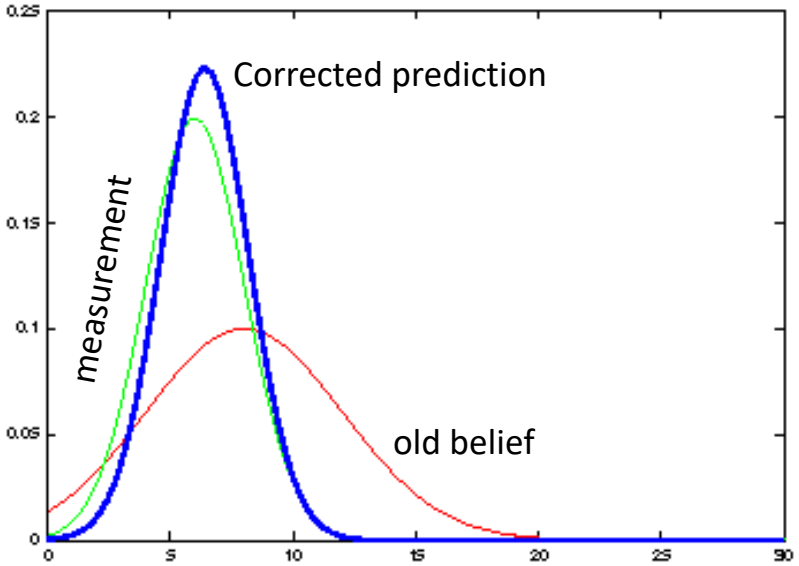
Measurement

Corrected prediction

Slide credit: Kristen Grauman

Tracking as inference: intuition





Time t Time t+1

Slide credit: Kristen Grauman

Independence assumptions

- Only immediate past state influences current state

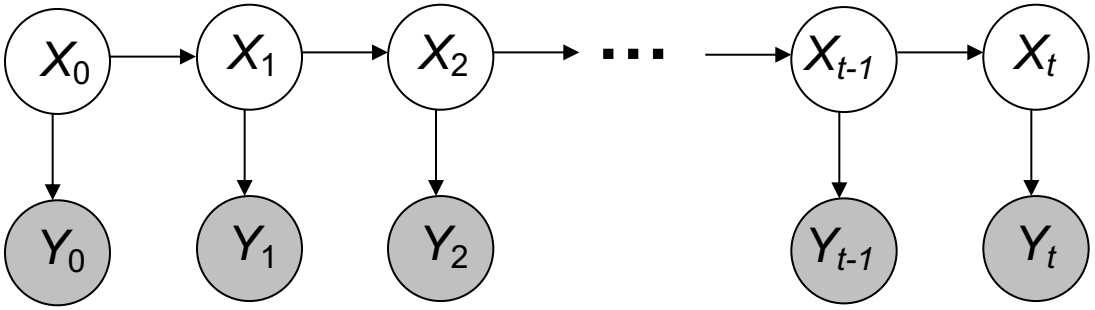
$$P(X_t|X_0,\ldots,X_{t-1}) = P(X_t|X_{t-1})$$

dynamics model

- Measurement at time t depends on current state

$$P(Y_t|X_0,Y_0\ldots,X_{t-1},Y_{t-1},X_t) = P(Y_t|X_t)$$

observation model



Hidden Markov model

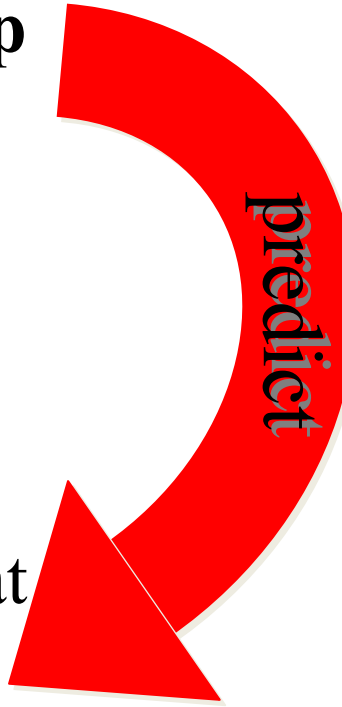
Tracking as inference

- Prediction:
  - Given the measurements we have seen **up to** this point, what state should we predict?

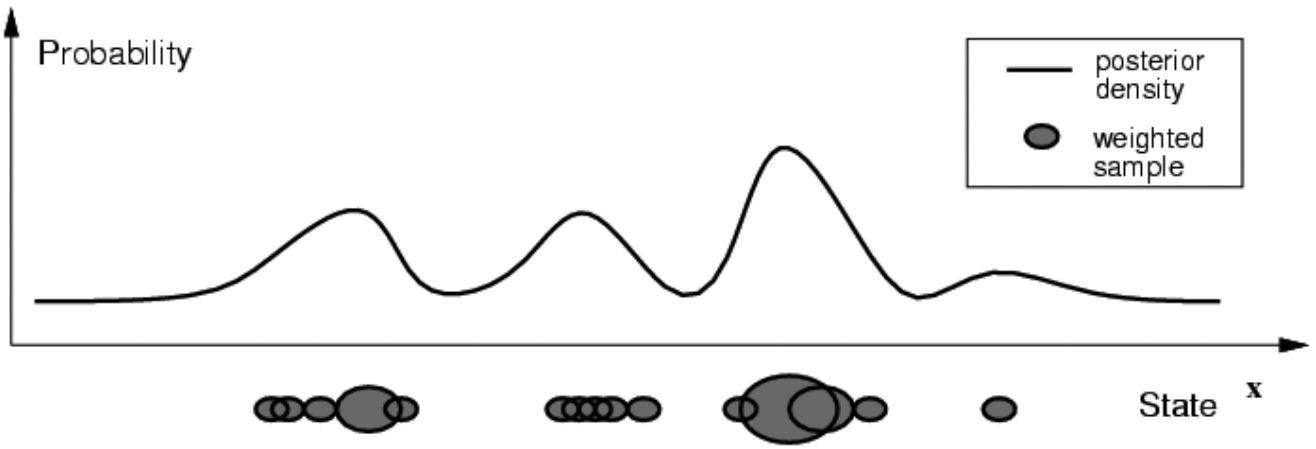
$$P(X_t | y_0, \ldots, y_{t-1})$$

- Correction:
  - Now given the **current** measurement, what state should we predict?

$$P(X_t | y_0, \ldots, y_t)$$



Particle filtering

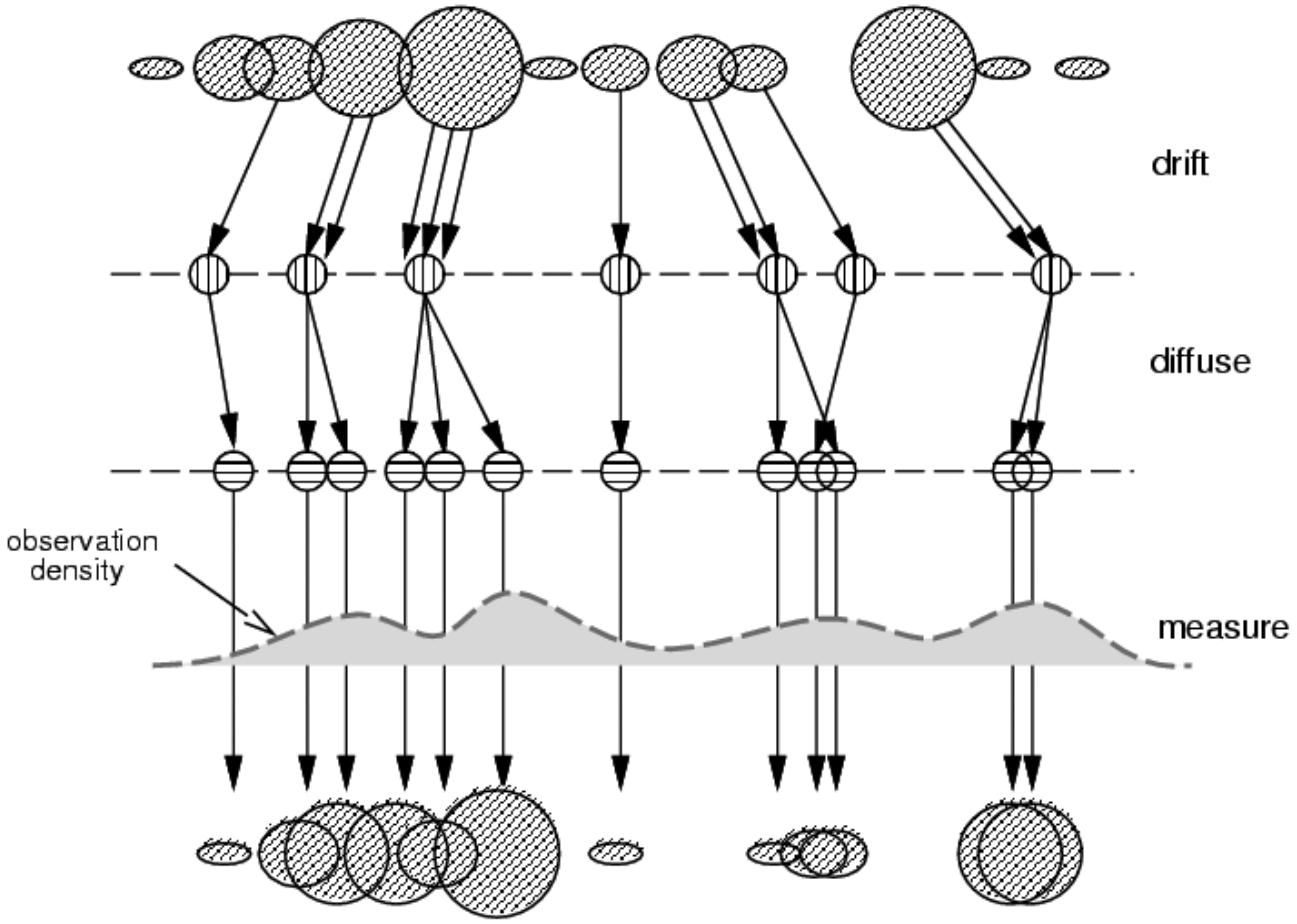


Represent the state distribution non-parametrically

- Prediction: Sample possible values X_{t-1} for the previous state and shift according to the dynamics model
- Correction: Compute likelihood of X_t based on weighted samples and $P(y_t|X_t)$

M. Isard and A. Blake, CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

Particle filtering



Start with weighted samples from previous time step

Sample and shift according to dynamics model

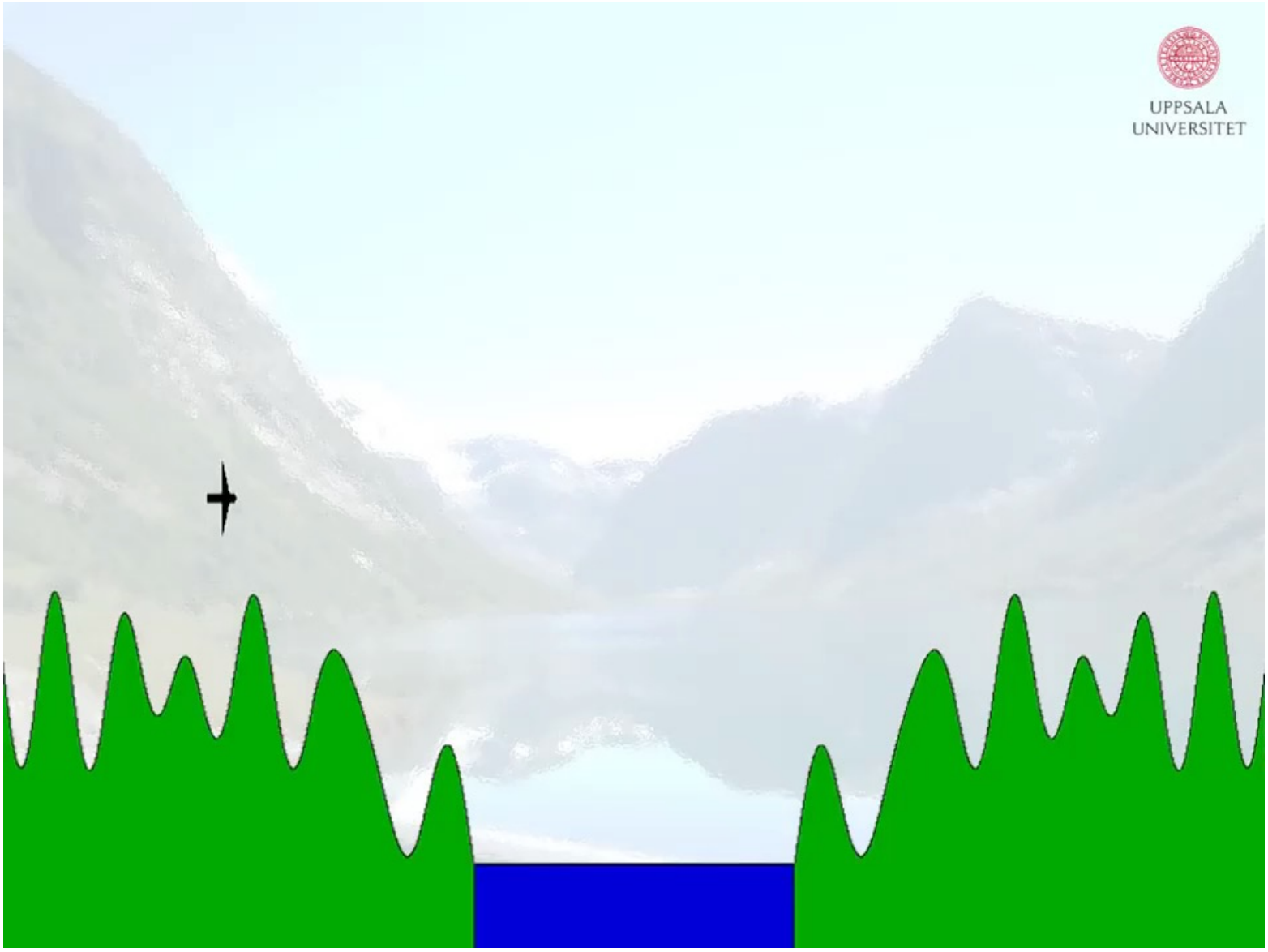
Spread due to randomness; this is predicted density $P(X_t|Y_{t-1})$

Weight the samples according to observation density

Arrive at corrected density estimate $P(X_t|Y_t)$

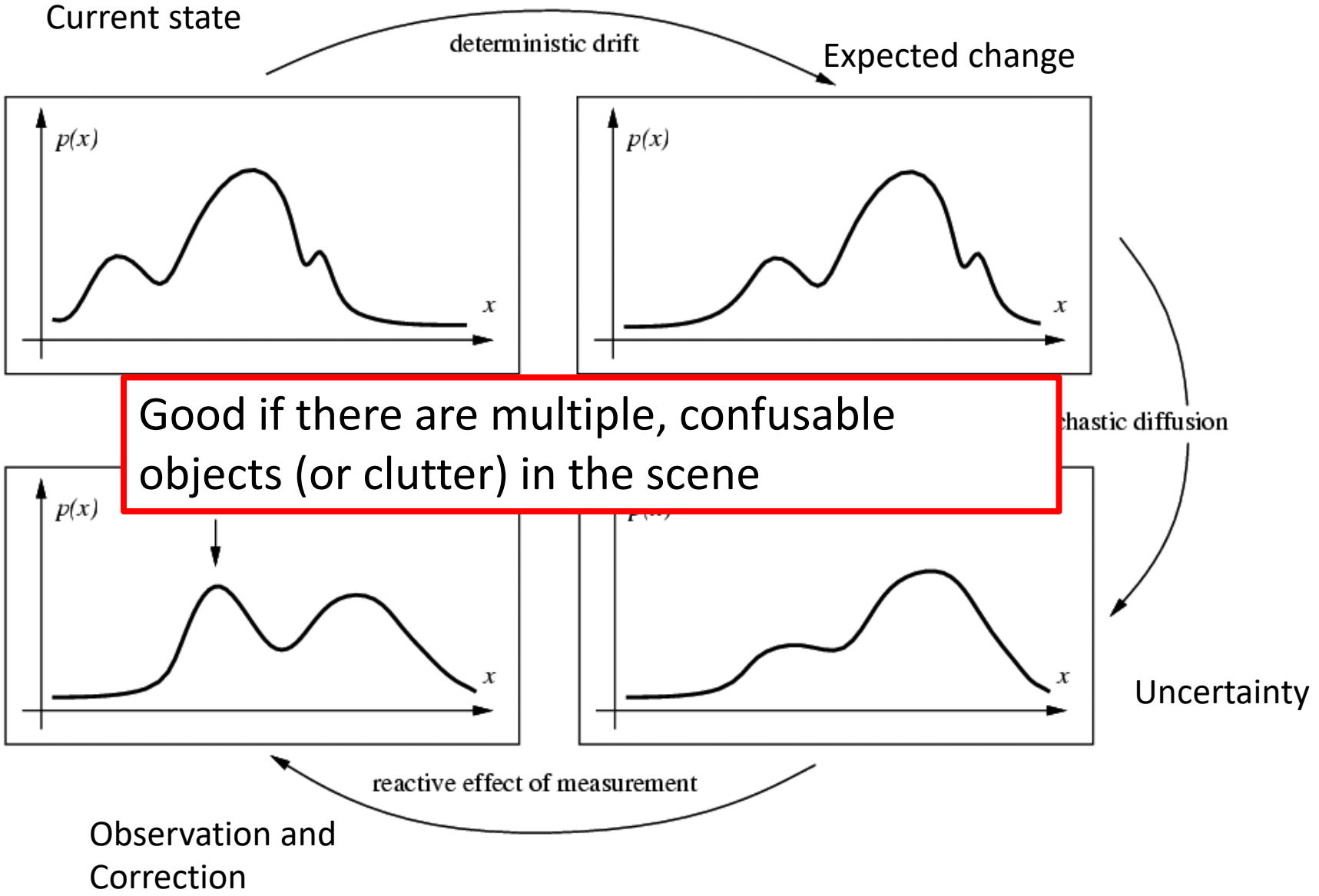
M. Isard and A. Blake, CONDENSATION -- conditional density propagation for visual tracking, IJCV 29(1):5-28, 1998

Particle filtering explained without equations

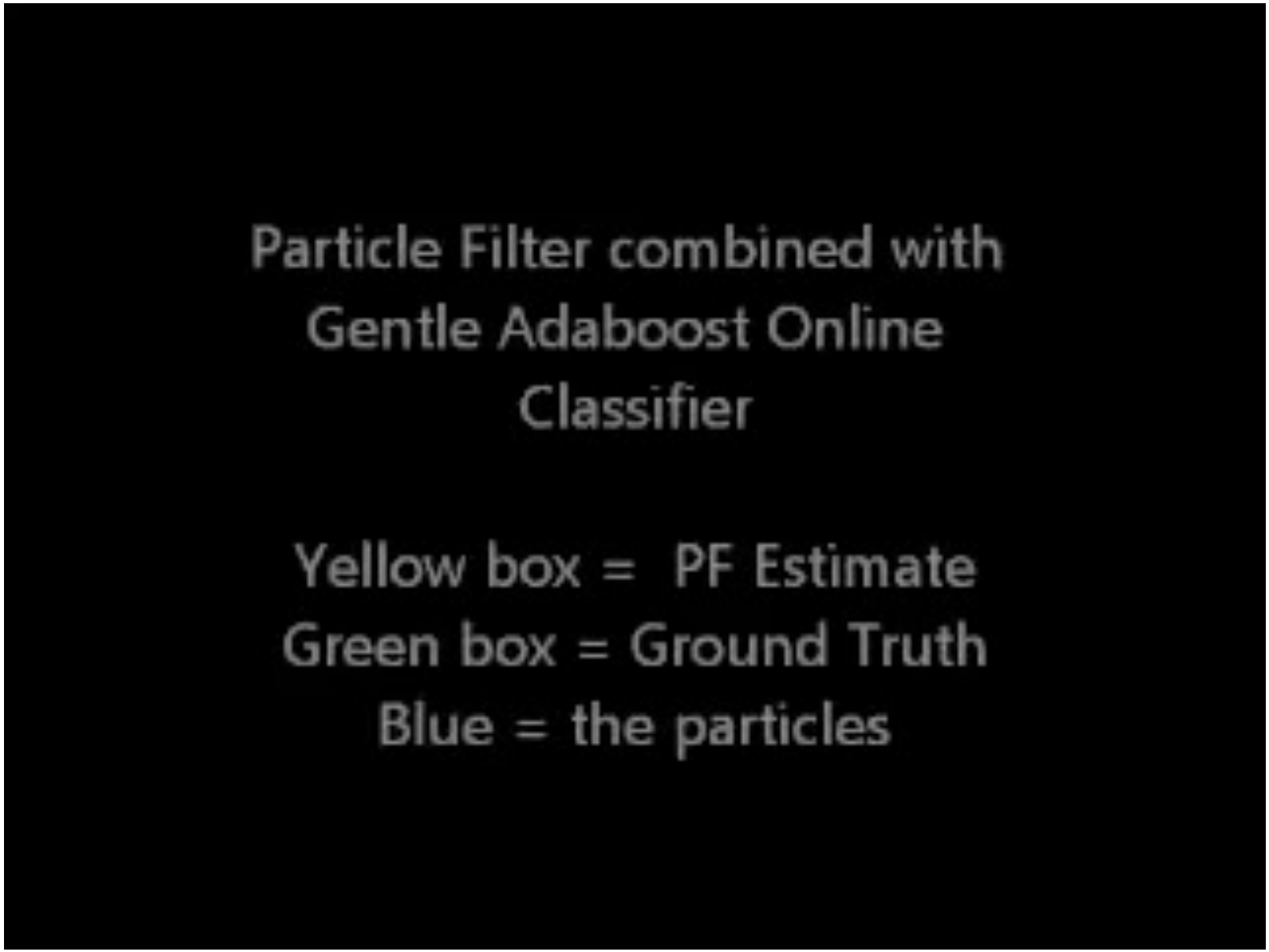


https://www.youtube.com/watch?v=aUkBa1zMKv4

Propagation of non-parametric densities



Particle filtering results



https://www.youtube.com/watch?v=O1FZyWz_yj4

Particle filtering results

https://www.youtube.com/watch?v=tIjufInUqZM

Tracking issues

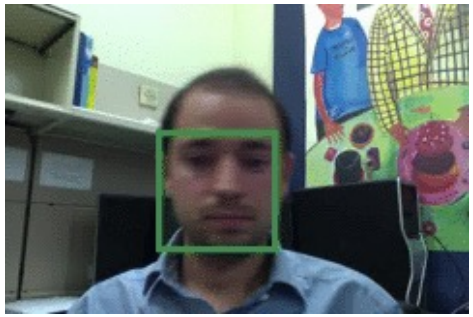
- Initialization
  - Manual
  - Background subtraction
  - Detection

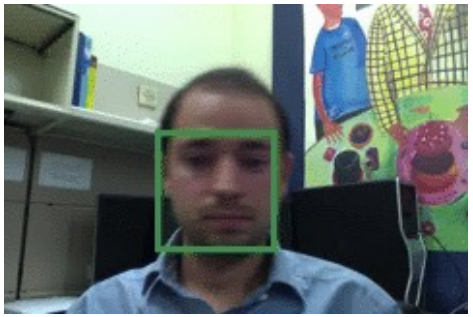
Tracking issues

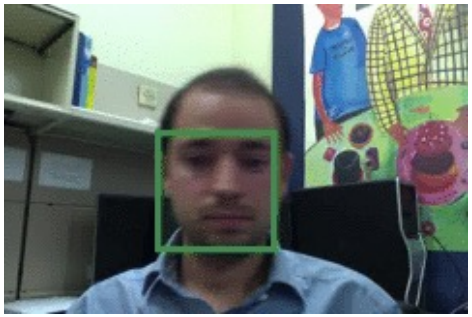
- Initialization
- Getting observation and dynamics models
  - Observation model: match a template or use a trained detector
  - Dynamics model: usually specify using domain knowledge

Tracking issues

- Initialization
- Obtaining observation and dynamics model
- Uncertainty of prediction vs. correction
  - If the dynamics model is too strong, will end up ignoring the data
  - If the observation model is too strong, tracking is reduced to repeated detection





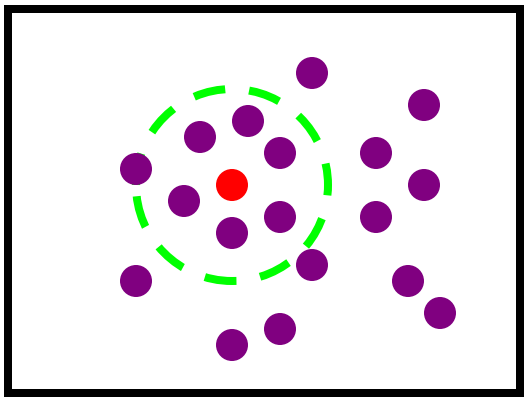


Too strong dynamics model

Too strong observation model

Tracking issues

- Initialization
- Getting observation and dynamics models
- Prediction vs. correction
- Data association
  - When tracking multiple objects, need to assign right objects to right tracks (particle filters good for this)



Tracking issues

- Initialization
- Getting observation and dynamics models
- Prediction vs. correction
- Data association
- Drift
  - Errors can accumulate over time

Drift

D. Ramanan, D. Forsyth, and A. Zisserman. Tracking People by Learning their Appearance. PAMI 2007.