Computer Vision

Lecture 1: Introduction

23.10.2024

Manuel Heurich, Tim Landgraf

Meet the team

Manuel

Chief of Assignments

Tim

Gives Lectures

https://bioroboticslab.github.io/website/

Assignments / Quizzes

- Assignments go deeper for selected key concepts in CV
  - Default: Jupyter notebooks, pull from git repo
  - Submission
    - in **groups** (max 2 students)
    - upload to Whiteboard as **PDF**
  - Collect **BONUS** points for quizzes
- 2 Quizzes instead of 1 Final
  - dates in KVV (don‘t forget to register!)
  - Approx. 45 min each
  - We‘ll grade **sum of points**, not average grades

List of Topics

SUBJECT TO CHANGE (Let me know!)

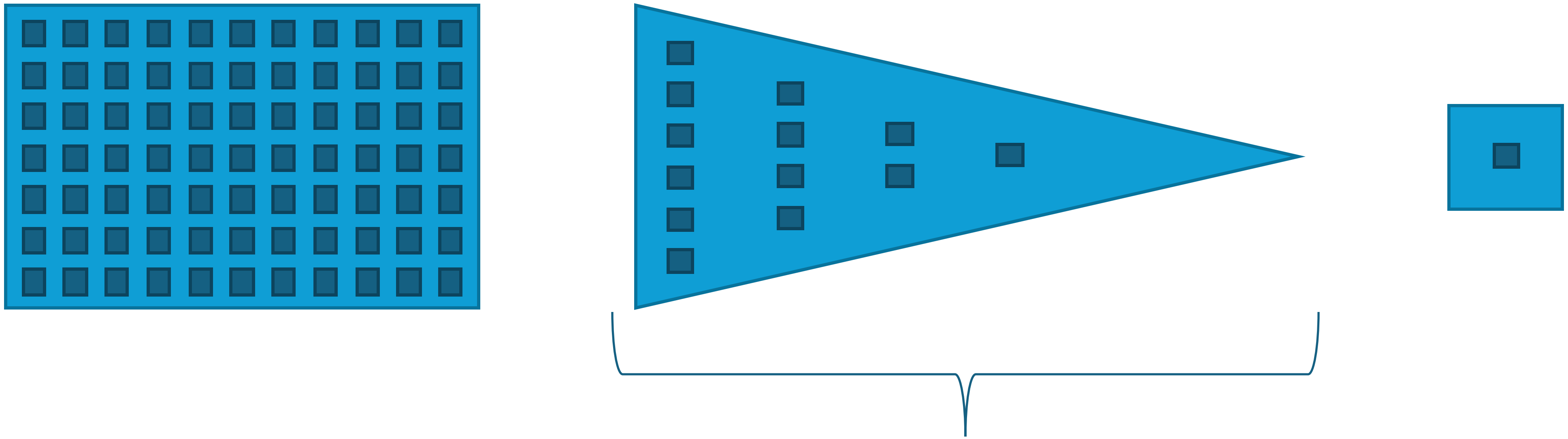
1.	Introduction	Computer Vision Fundamenta
2.	Edge Detectors, Convolution	
3.	Color Histograms, HoG	
4.	Optic Flow	Conventional Computer Vision
5.	Hough Transform	
6.	SIFT / SURF	
7.	Introduction to Neural Information Processing	Deep Learning (Supervised)
8.	Convolutional Neural Networks	
9.	Image Classification, Object Detection	
10.	Vision Transformers, ConvMixer	
11.	Semantic Segmentation	
12.	Pose Estimation	
13.	Recurrent Neural Networks, Image Captioning	
14.	Contrastive Learning	Self-Supervised Learning
15.	BYOL, IJEPA, VICREG	

What is „Computer Vision“

High-dimensional,
raw pixel data

Relevant Features

Low-dimensional,
abstract information



FEATURE EXTRACTION

What is „Computer Vision“

In

https://blog.alldogboots.com/wp-content/uploads/2010/02/SmilingDog.gif

Out

- dog is in / in not in image
- „dog“, „outside“, „smile“, „snow“
- bounding_box(400, 234, 200, 150)
- position(t)
- „two huskys on ice ...“

What is „Computer Vision“

In

https://blog.alldogboots.com/wp-content/uploads/2010/02/SmilingDog.gif

Out

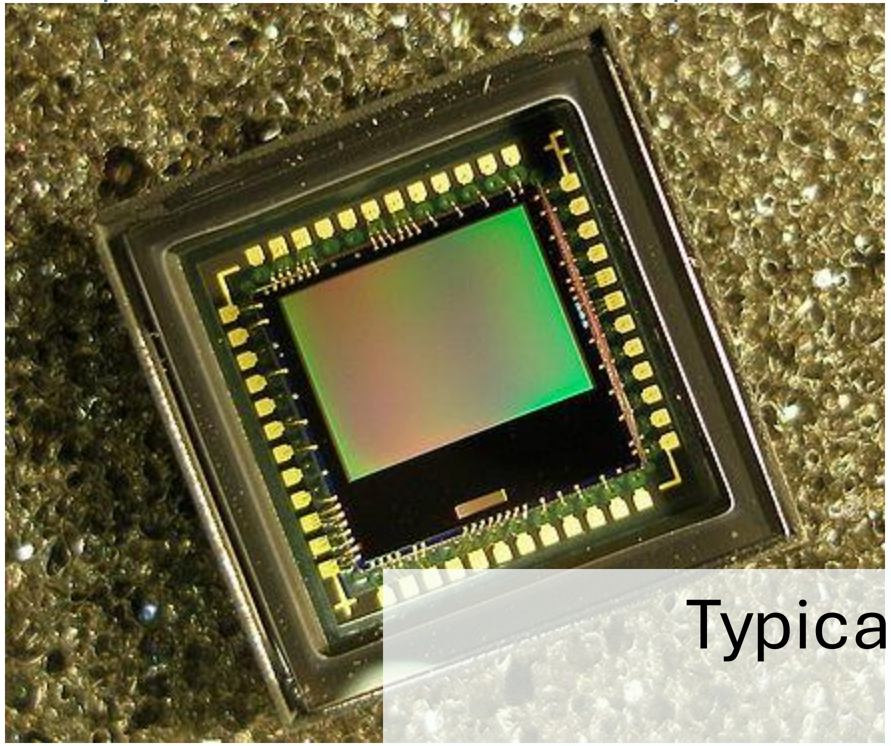
- detection
- classification
- localisation
- tracking
- captioning
- segmentation

How do we represent digital images?

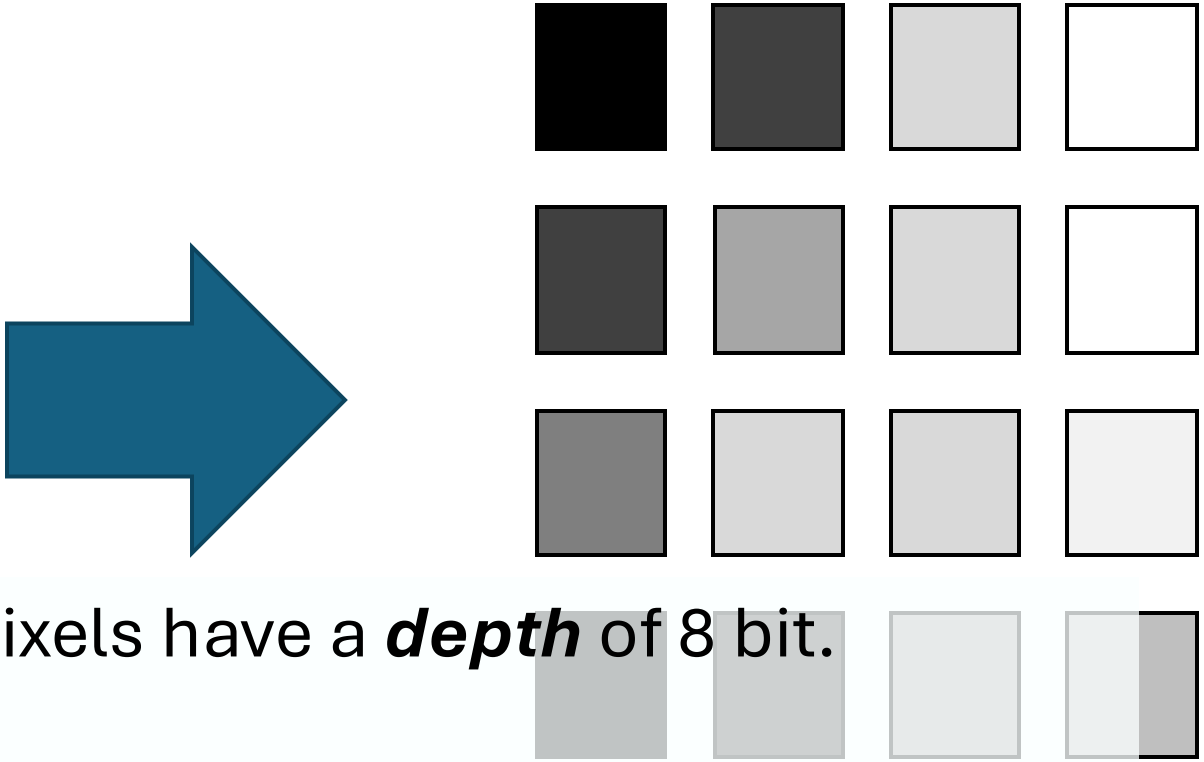
https://www.alphapaw.com/blog/smiling-dog-memes/

Grayscale Images

https://www.wikiwand.com/en/Active-pixel_sensor



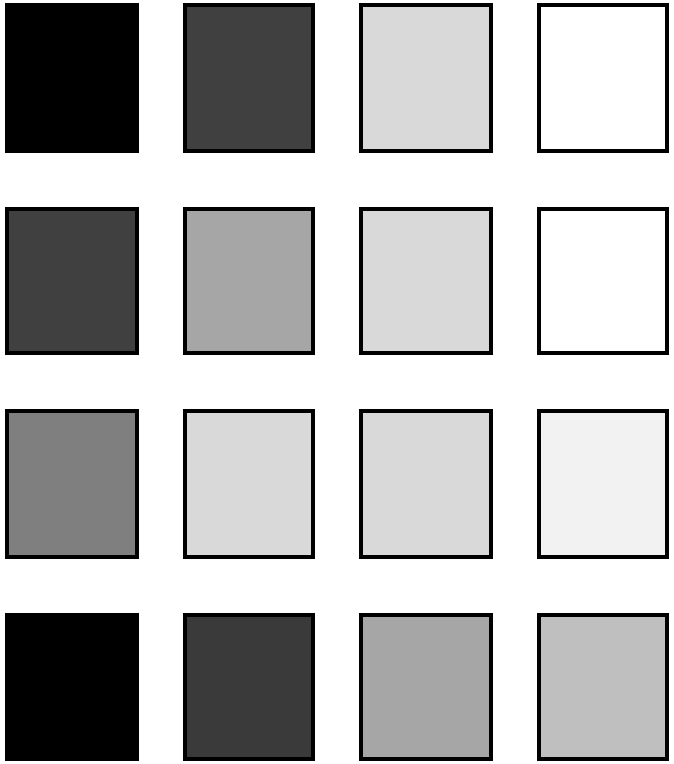
CMOS senso



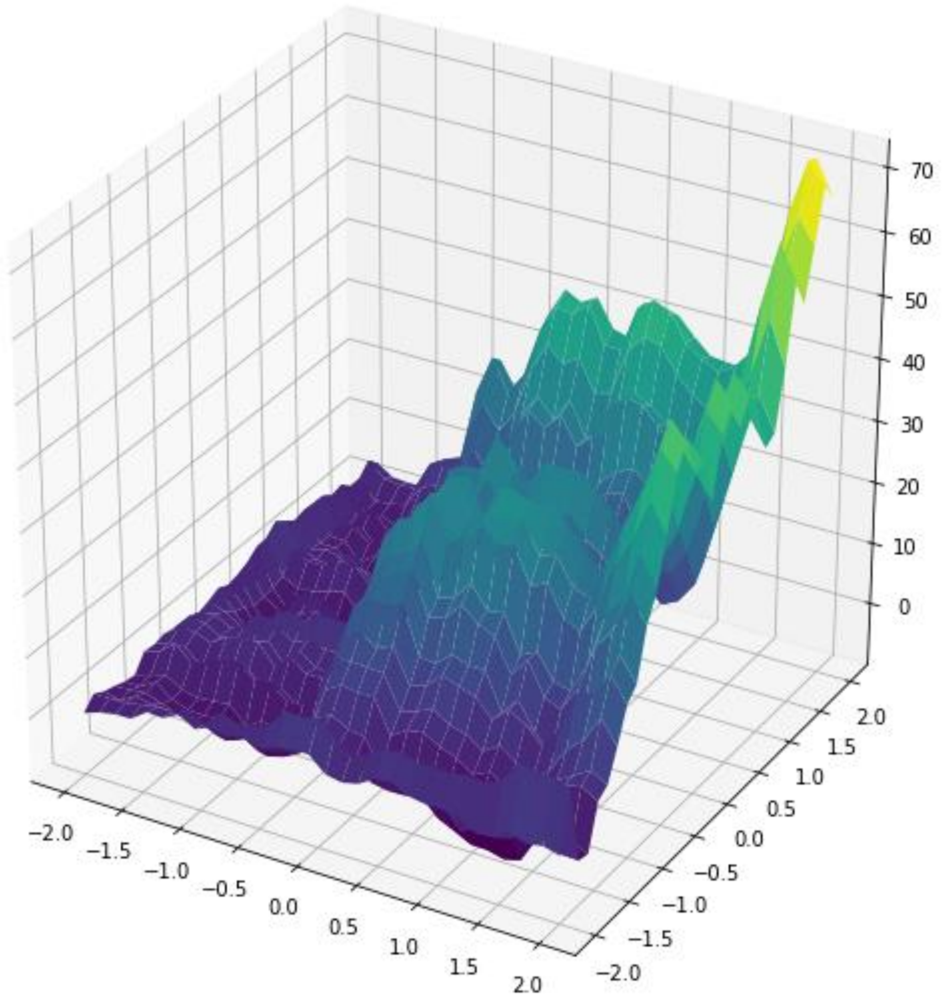
Typically, pixels have a ***depth*** of 8 bit.

This corresponds to **how many different gray values**?

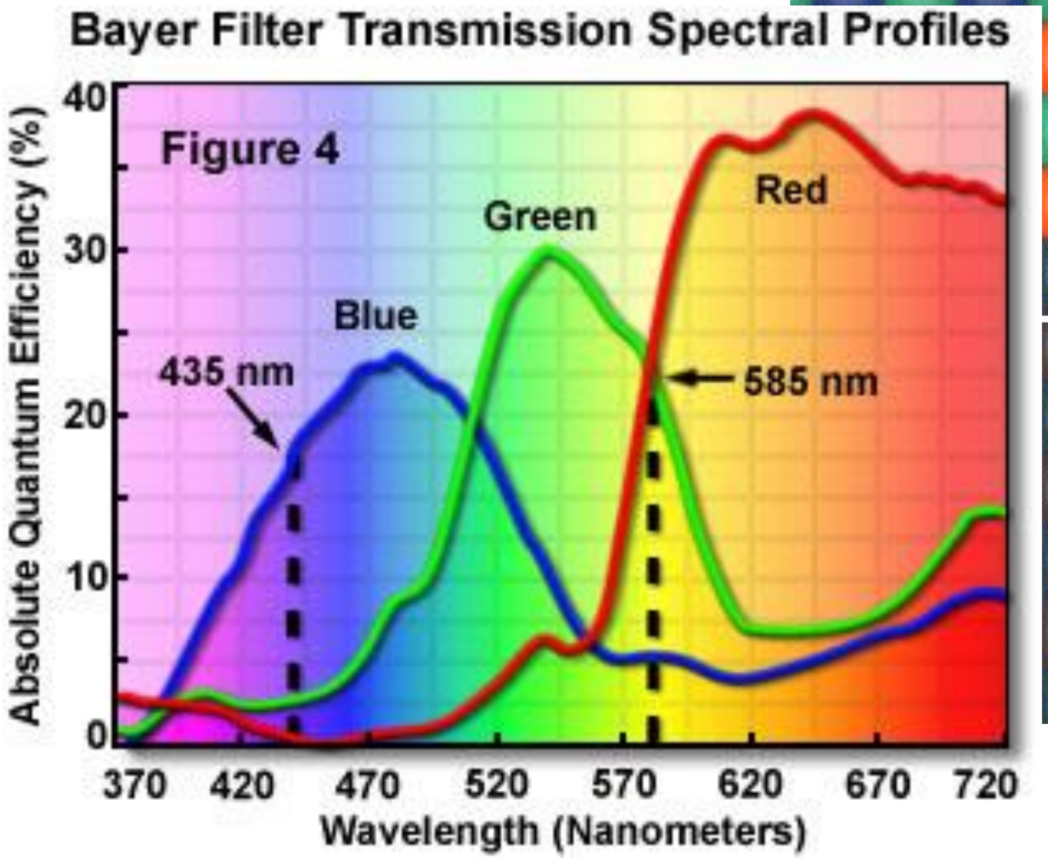
Grayscale Images



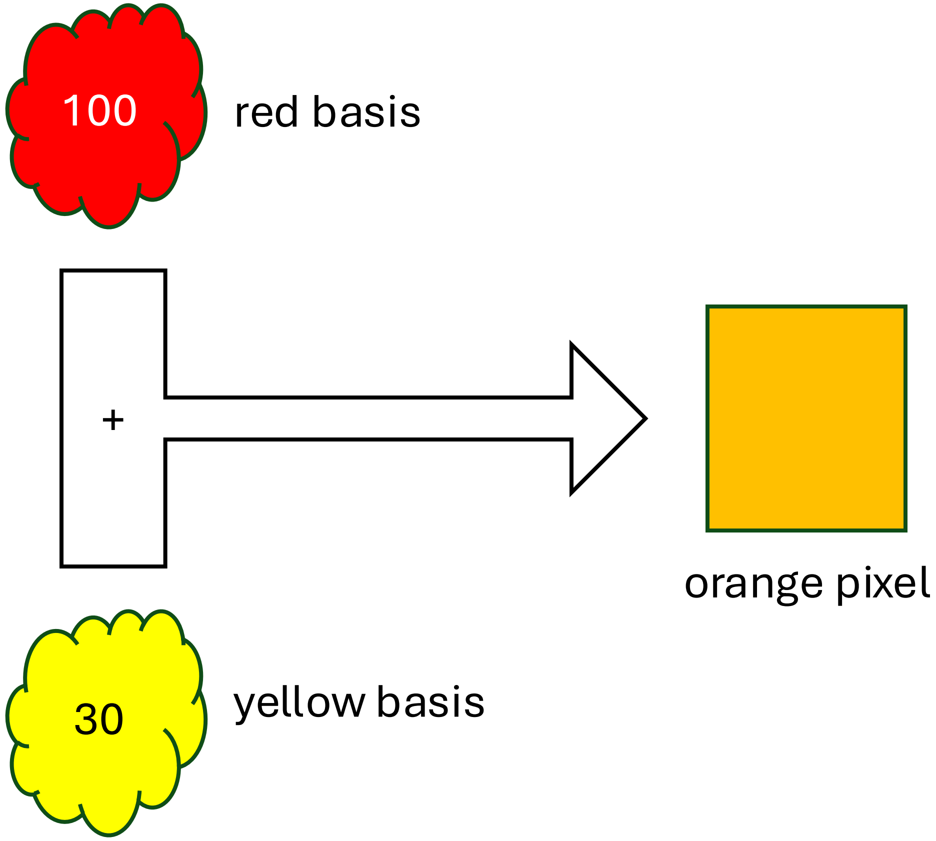
↔

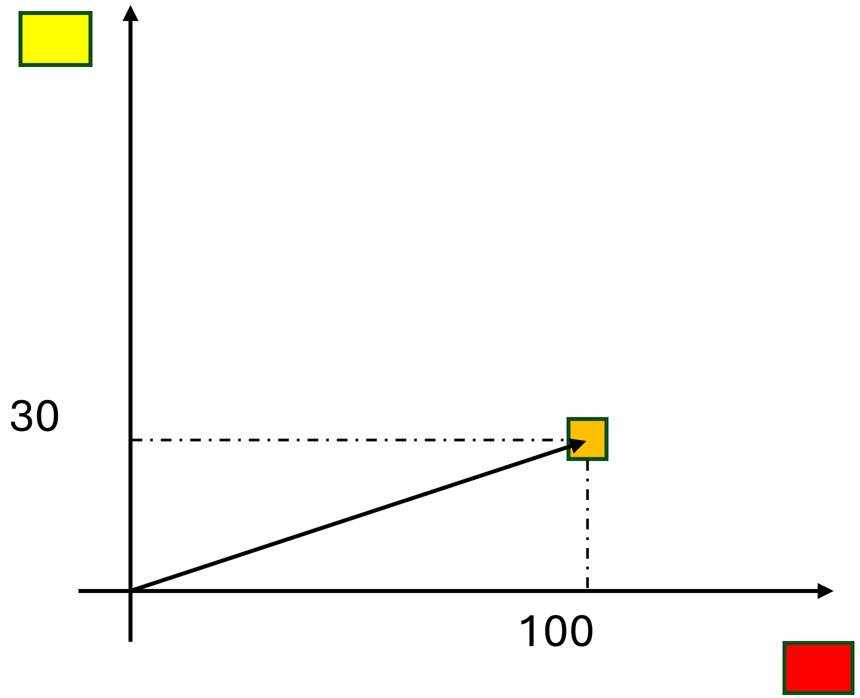


Color Images

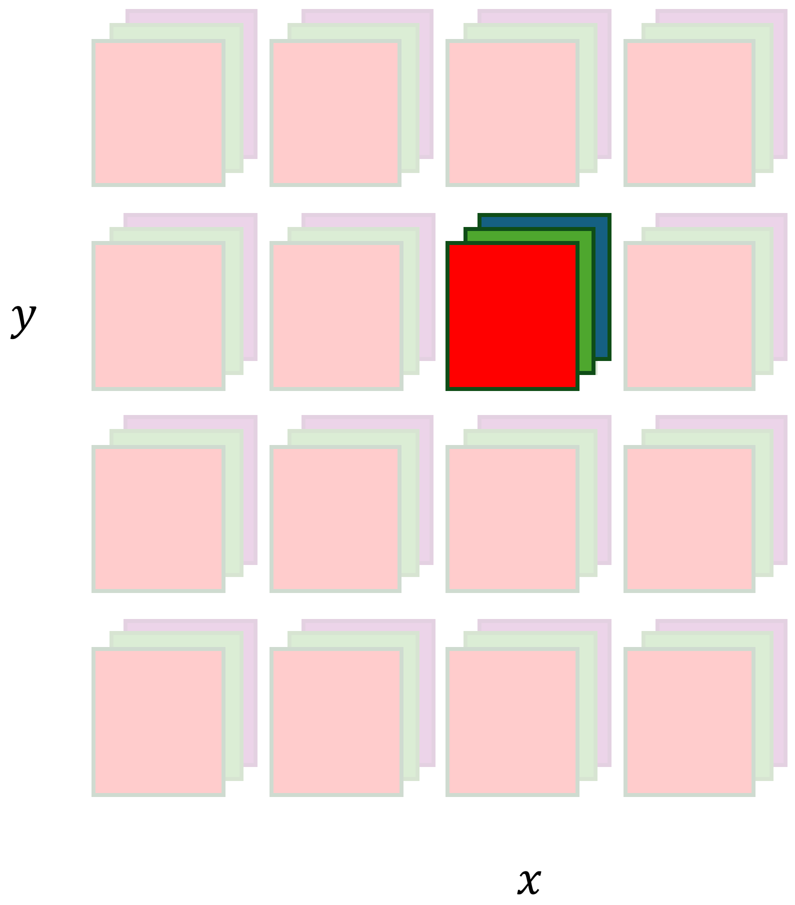


Color Images

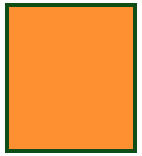




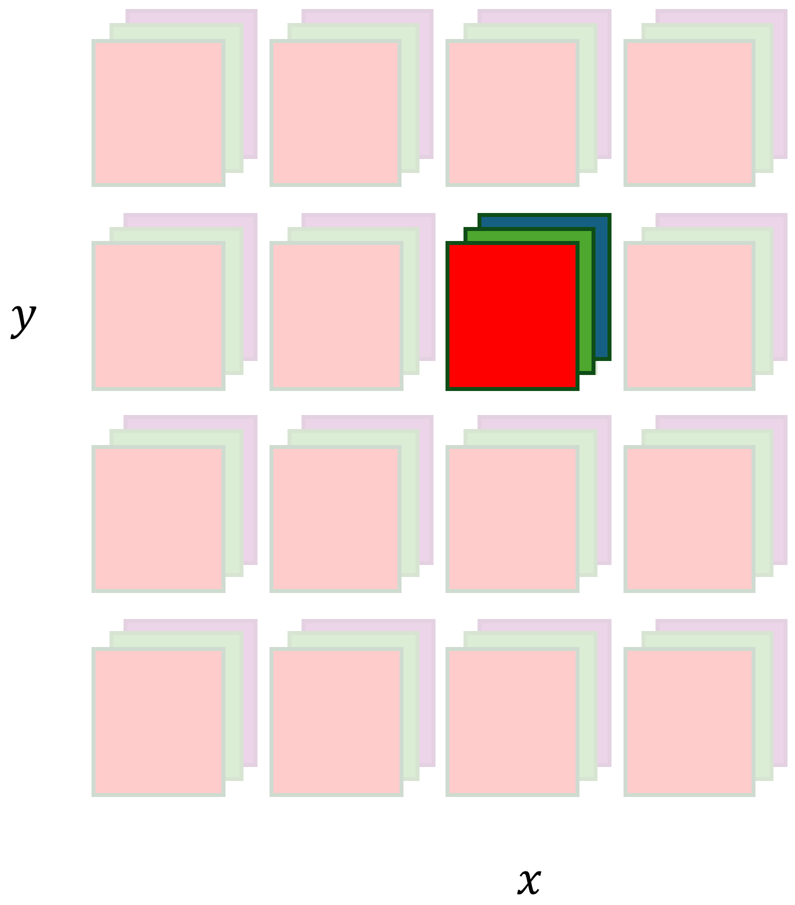
Color Images: RGB color space

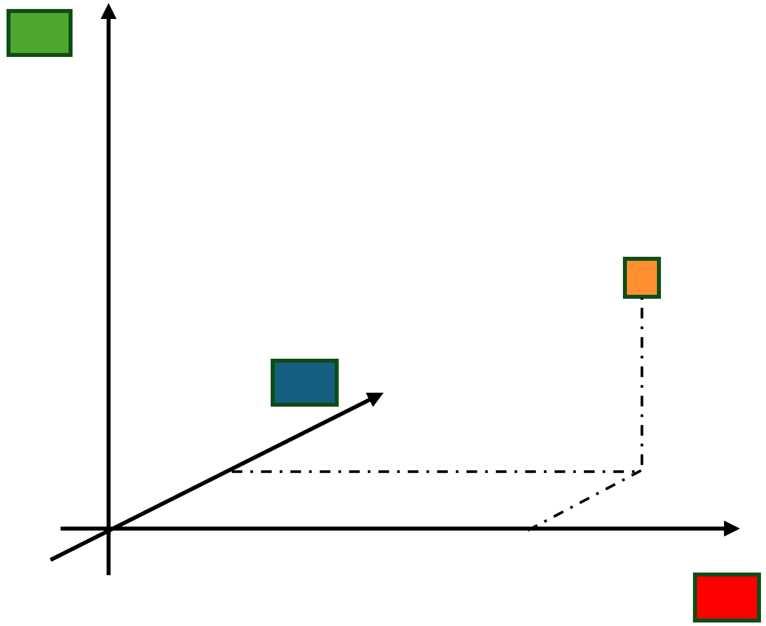


$$I(x, y) = \begin{pmatrix} 255 \\ 144 \\ 50 \end{pmatrix}$$



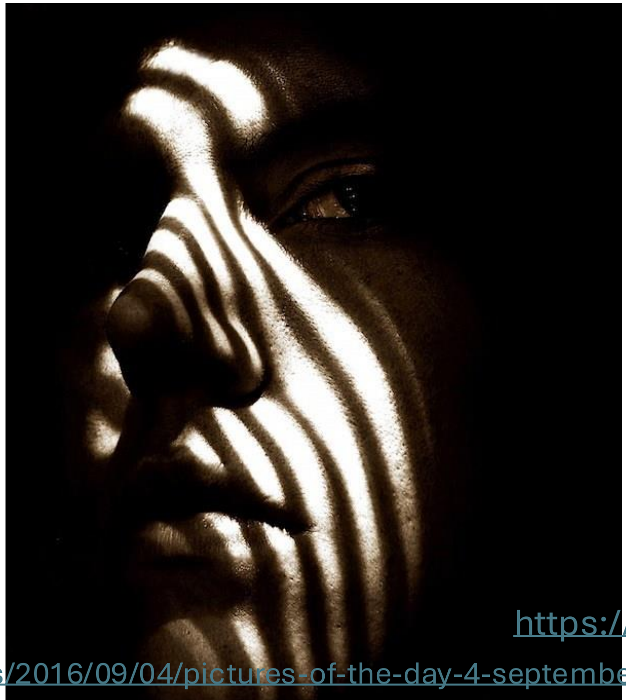
Color Images: RGB color space





https://slideplayer.com/slide/9414115/

- same-object variability
  - pose
  - shape
  - scale
  - motion
- intra-class variability
- between-class discriminability
- environmental variability
  - lighting
  - occlusion



https://www.pinterest.de/pin/281404676703747451/?d=t&mt=login

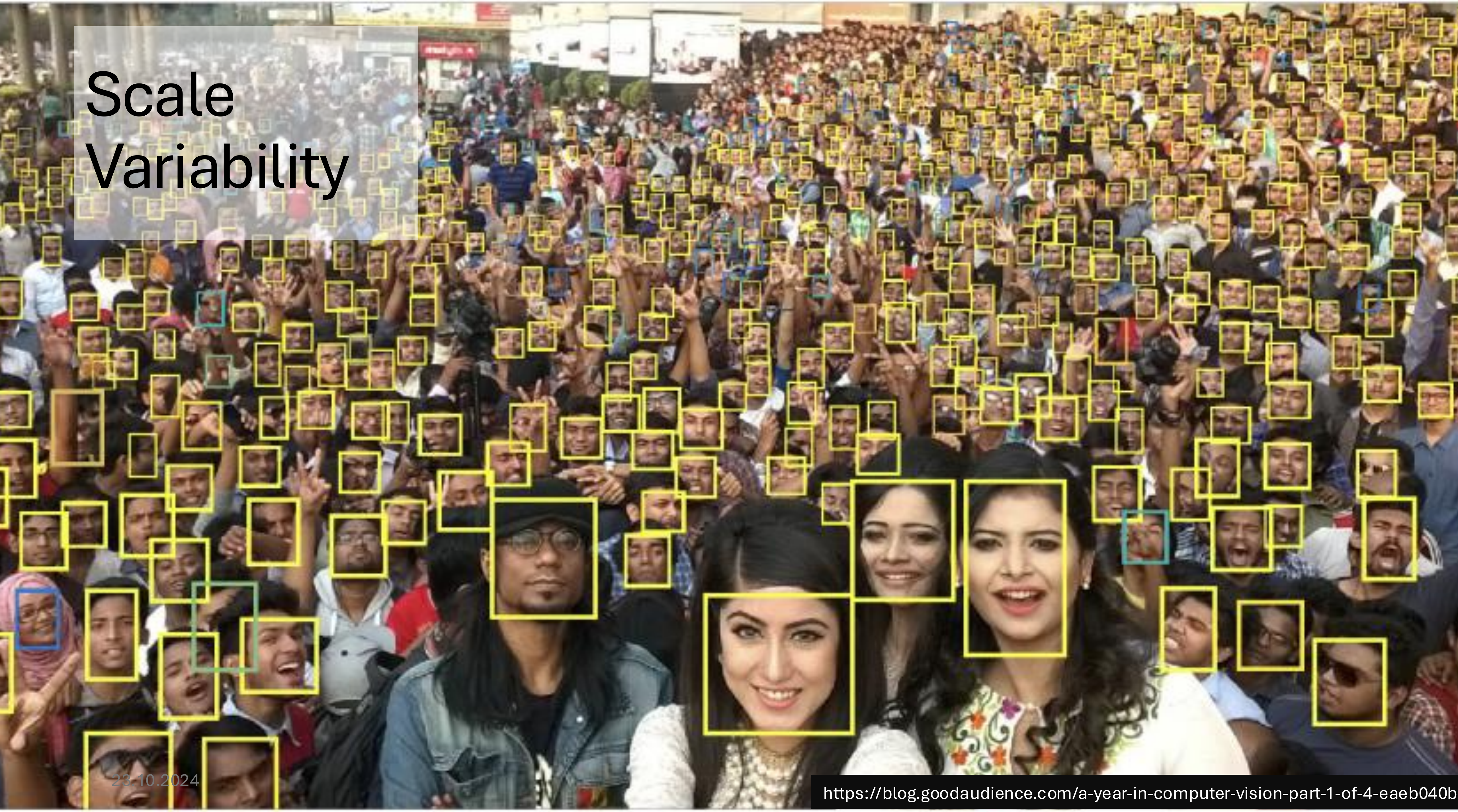
https://www.telegraph.co.uk/news/2016/09/04/pictures-of-the-day-4-september-20162/a-bear-plays-peekaboo-with-photographer-jari-peltomaki-in-a-

Between-class dicriminability

https://www.pinterest.de/pin/5840674486958115/

https://www.theguardian.com/uk-news/gallery/2015/jul/20/more-than-human-fine-animal-portraiture

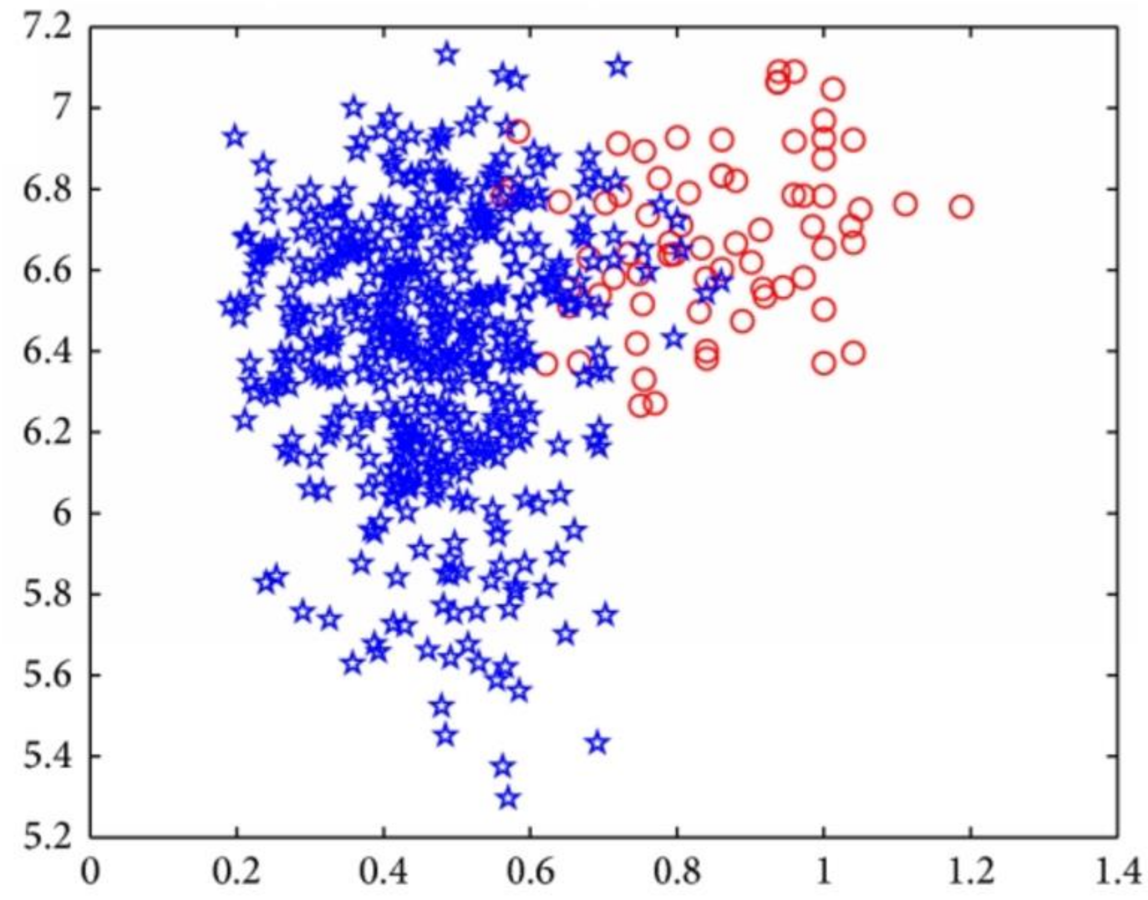
Scale Variability



https://blog.goodaudience.com/a-year-in-computer-vision-part-1-of-4-eaeb040b

Computer Vision: „what are **relevant** features?“

- comparing images on the level of pixel values?
- edges, color, motion, …
- a huge part of „conventional“ computer vision was manually defining what *relevant* means
- since computer vision has become „deep“: features are *learned*

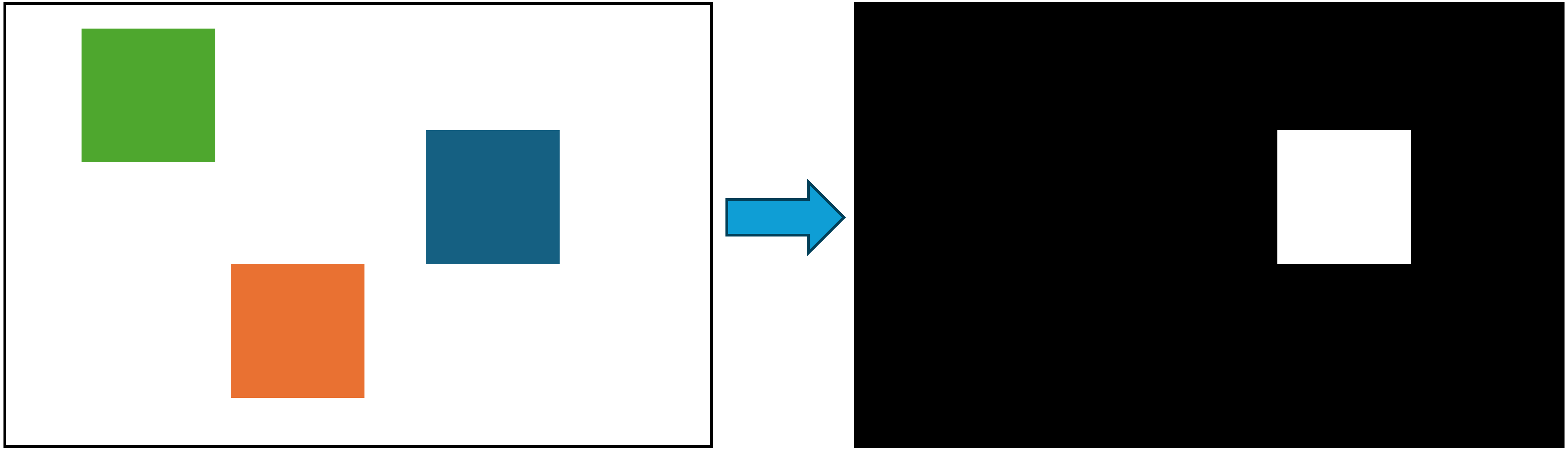


Color-based detection and tracking

- What is color?
- How can we use color to identify objects?
- How to represent foreground objects in color sub-spaces?
- What is the problem with color?

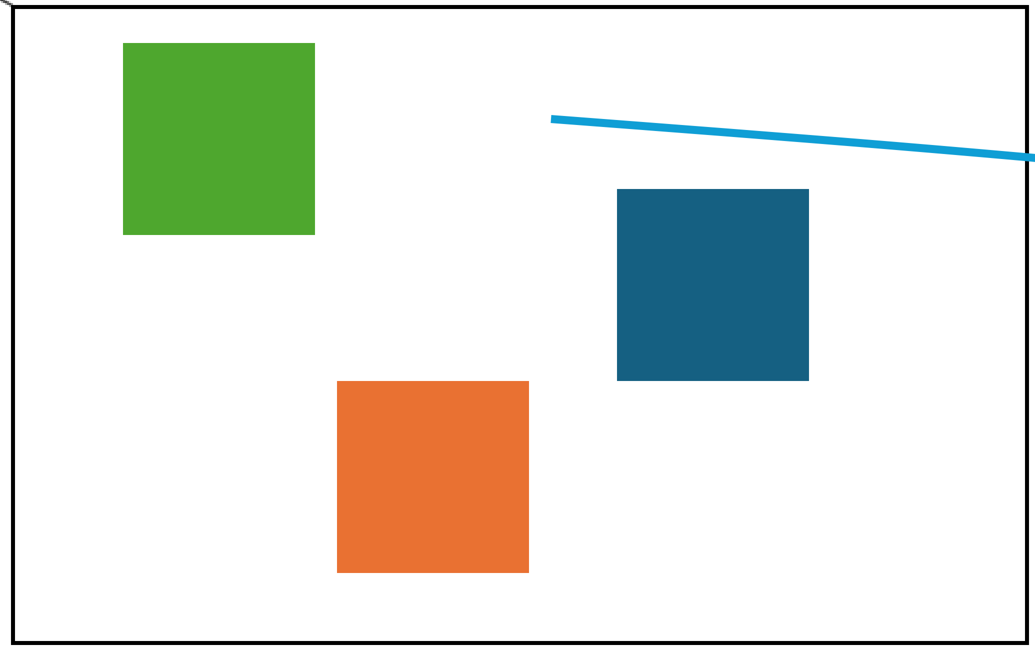
Detection via Binarization

„look for blue objects“

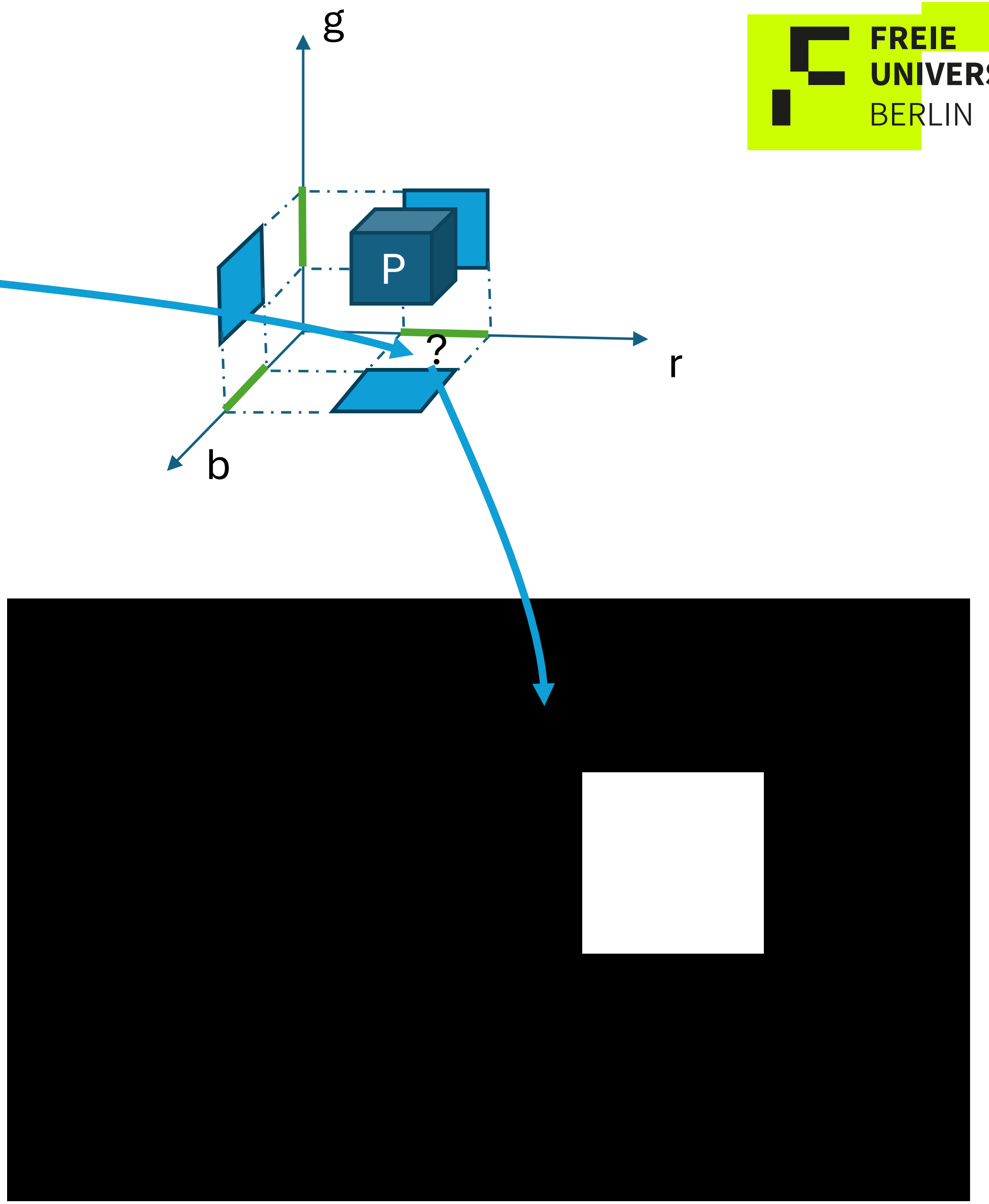


Colored objects in input image

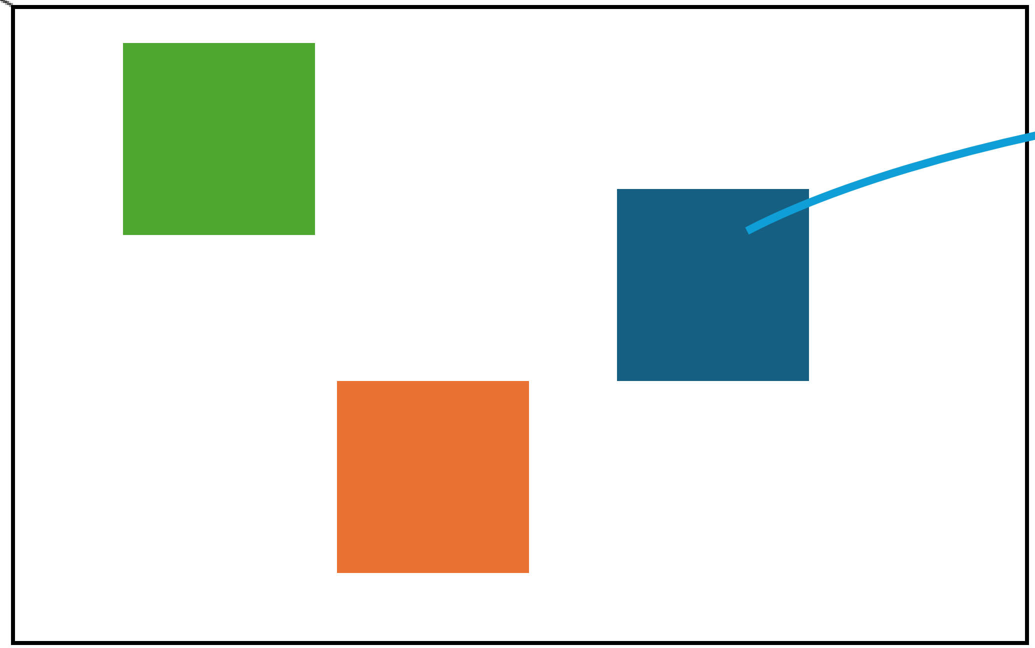
Binarization based on target color



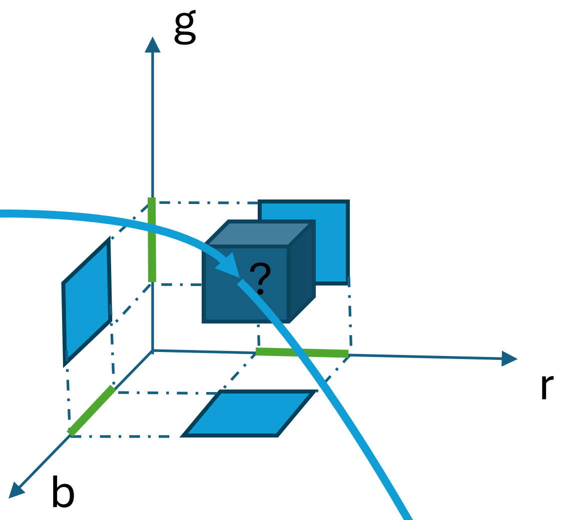
Colored objects in input image

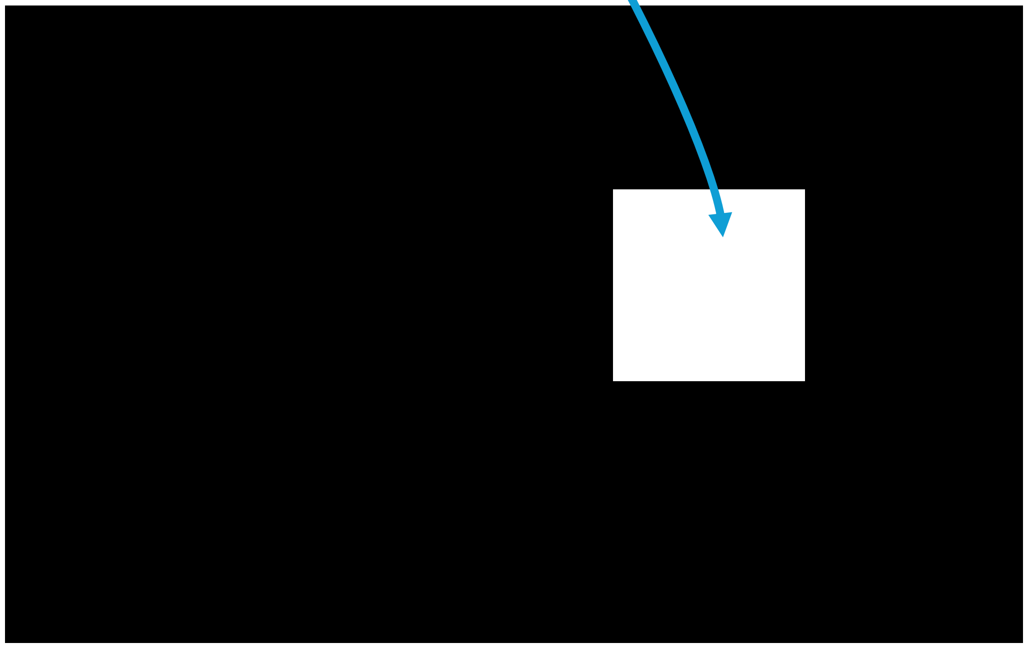


Binarization based on target color



Colored objects in input image

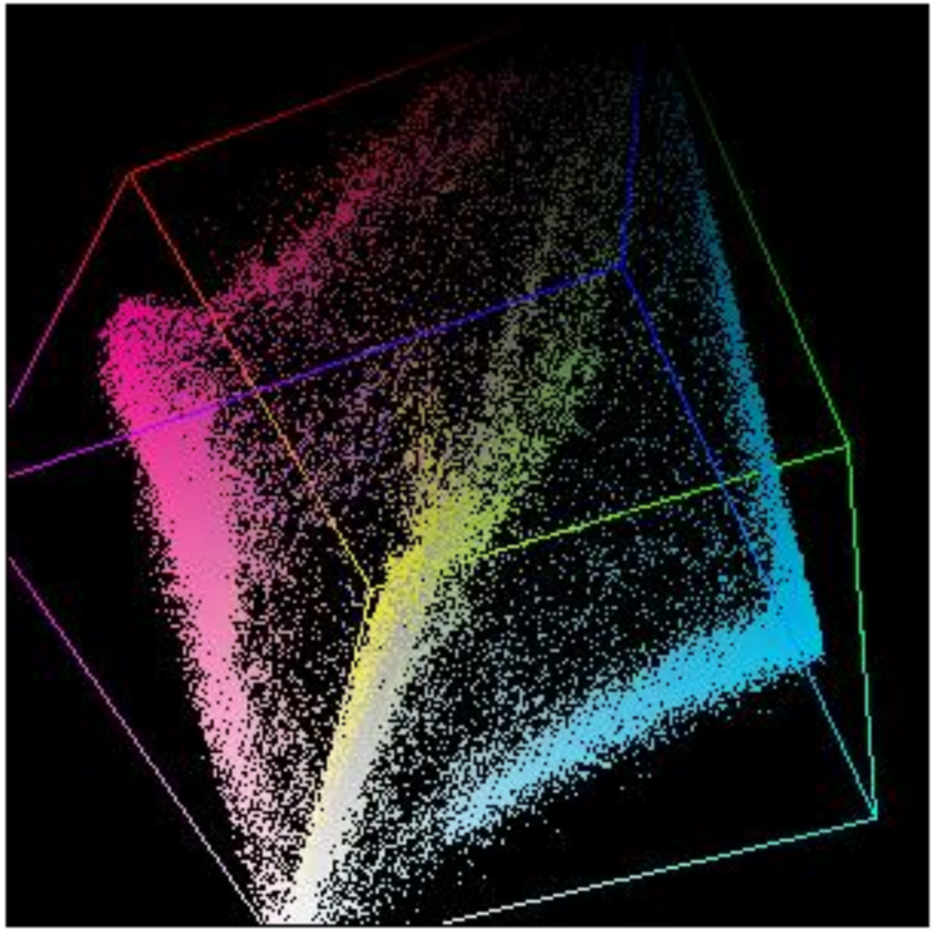




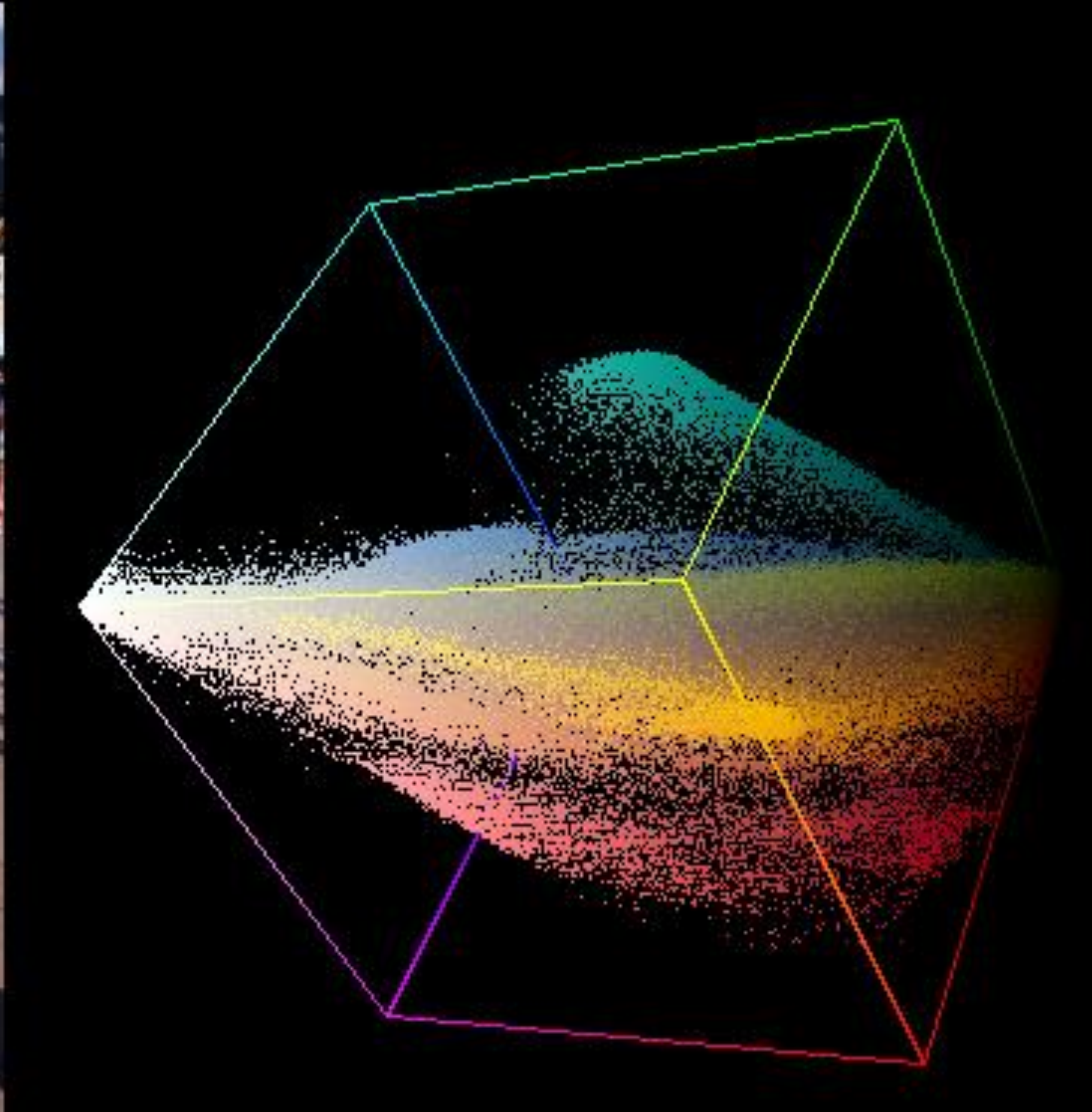
Binarization based on target color

Color-based detection and tracking

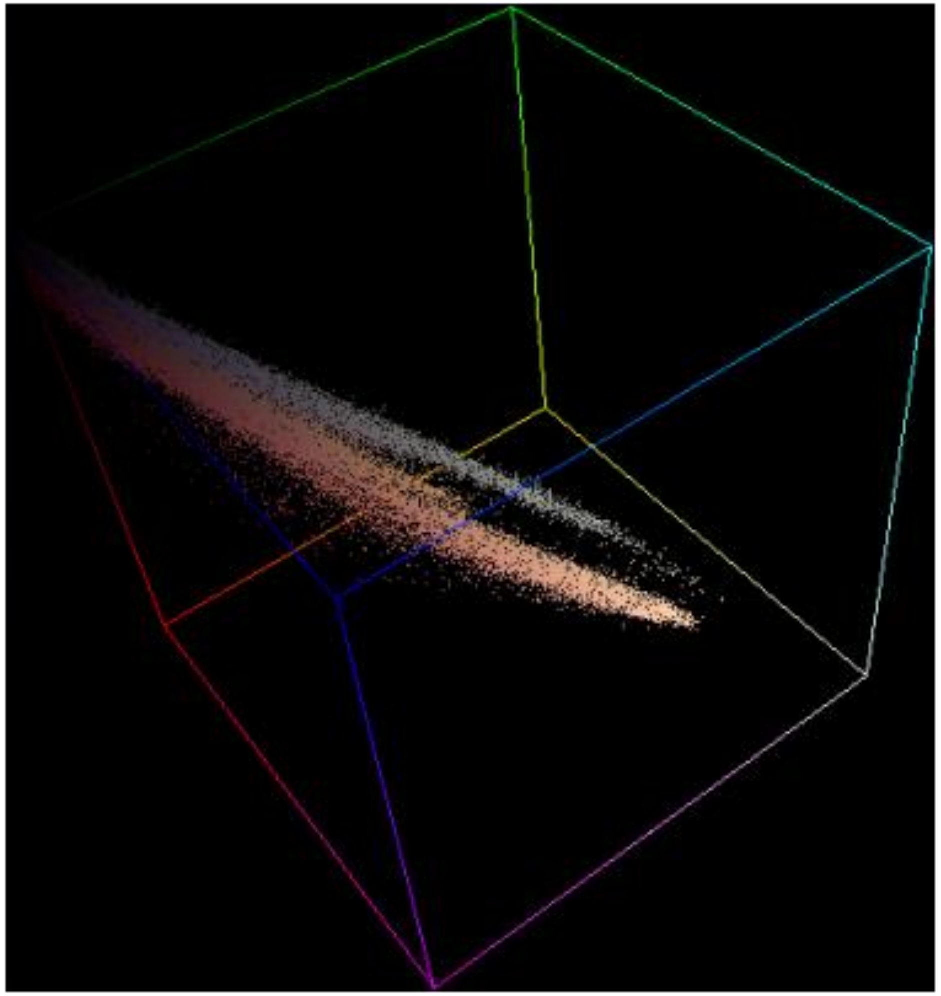
Are objects identifyable by color?



http://opensource.graphics/visualizing-the-3d-point-cloud-of-rgb-colors/



http://opensource.graphics/visualizing-the-3d-point-cloud-of-rgb-colo



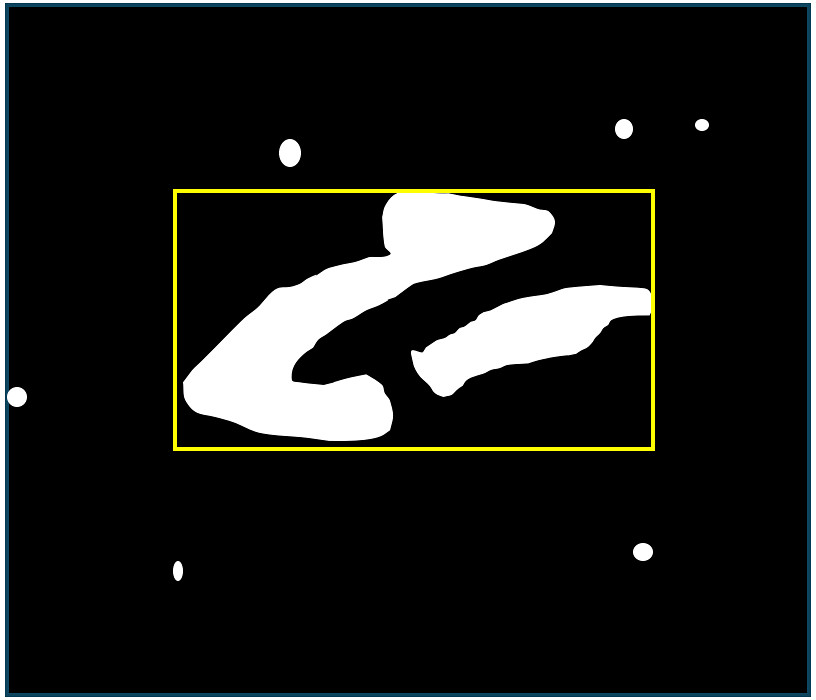
Are objects identifyable by color?

Ex. 1.1: detect colored objects in RGB

- define the „positive“ subspace P in the RGB cube
- iterate over all pixels in I and check if in P or ~P
- write result to new image
- play around with size and shape of P
  and display binary image

Finding objects

- Binary image still unhandy
- We need to extract the object‘s location
- How do we **represent** the object?
  - position
  - orientation
  - scale
  - spatial extent in image
  - …



Finding objects

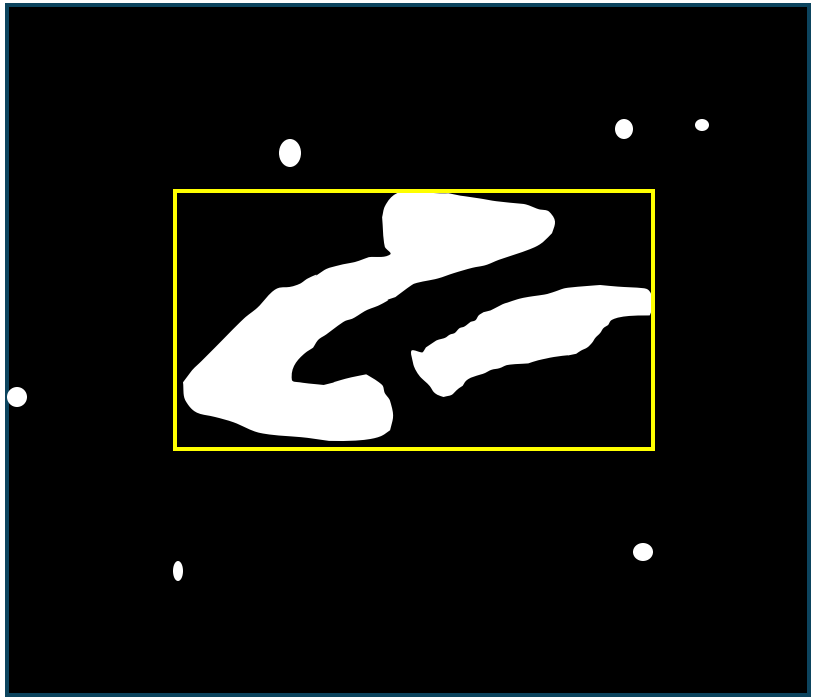
- Common receipe:
  1. apply **morphological operators**

     →remove noise

     →close gaps
  2. find **connected components**

     →iterate over all pixels

     →group pixels

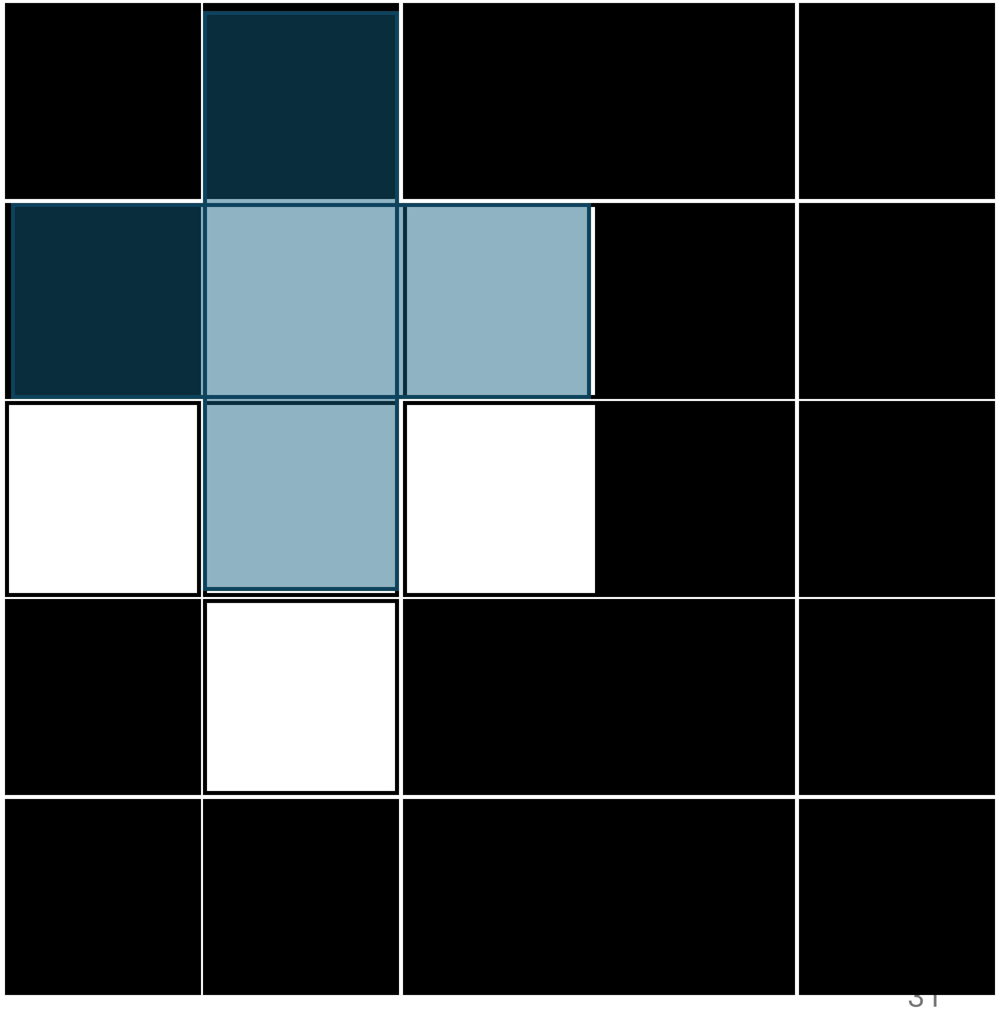


Morphological operat**ors**

- **Structuring Element** defines neighborhood around input pixel
- Set output pixel to „1“ if
  - all neighborpixels are „1“ (**Erosion**)
  - at least one of them is „1“ (**Dilation**)

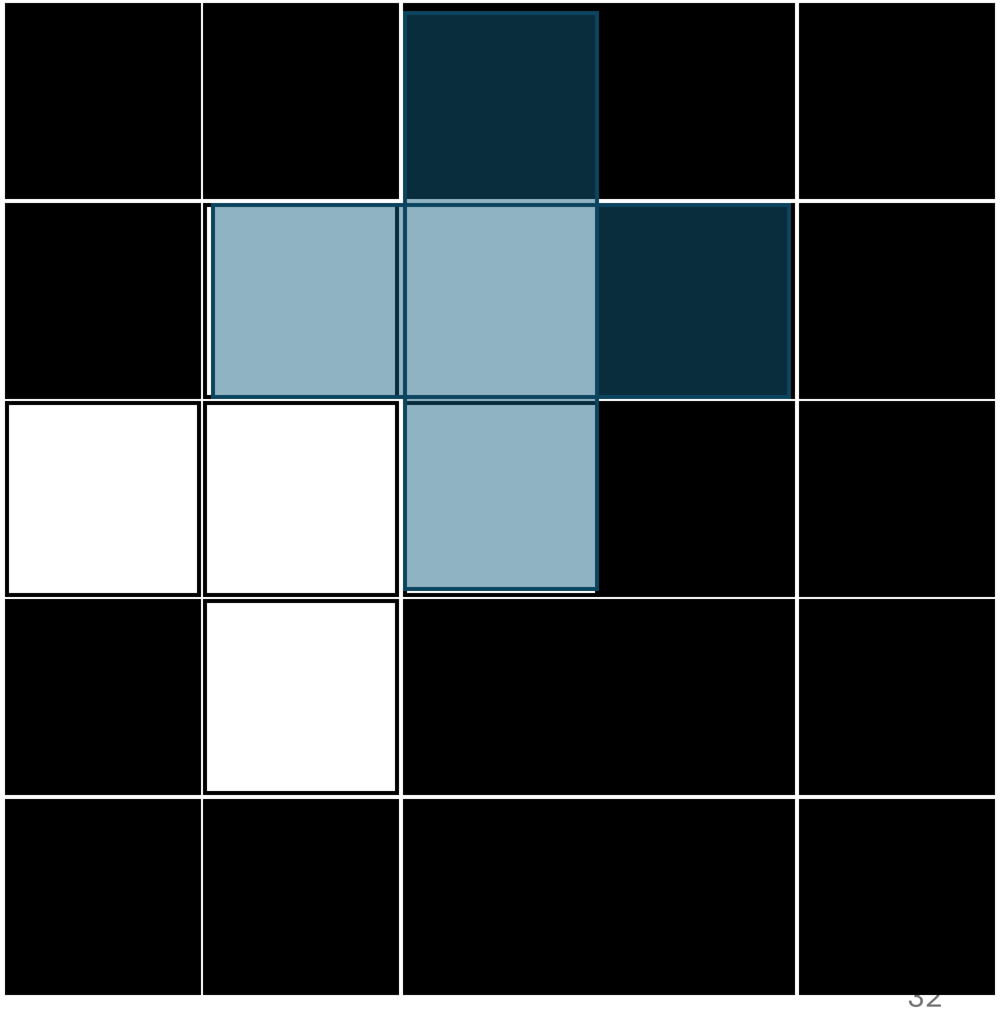
Morphological operat**ors**

- **Structuring Element** defines neighborhood around input pixel
- Set output pixel to „1“ if
  - all neighborpixels are „1“ (**Erosion**)
  - at least one of them is „1“ (**Dilation**)



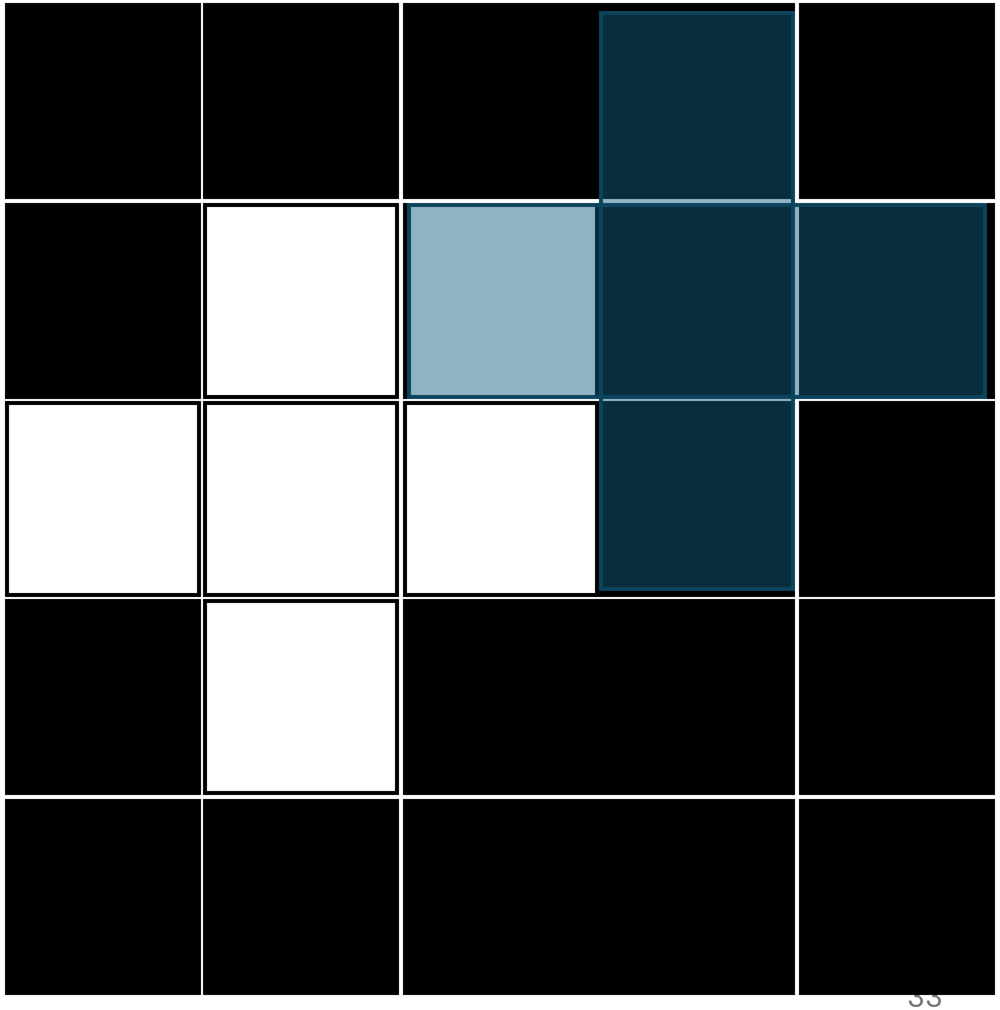
Morphological operat**ors**

- **Structuring Element** defines neighborhood around input pixel
- Set output pixel to „1“ if
  - all neighborpixels are „1“ (**Erosion**)
  - at least one of them is „1“ (**Dilation**)



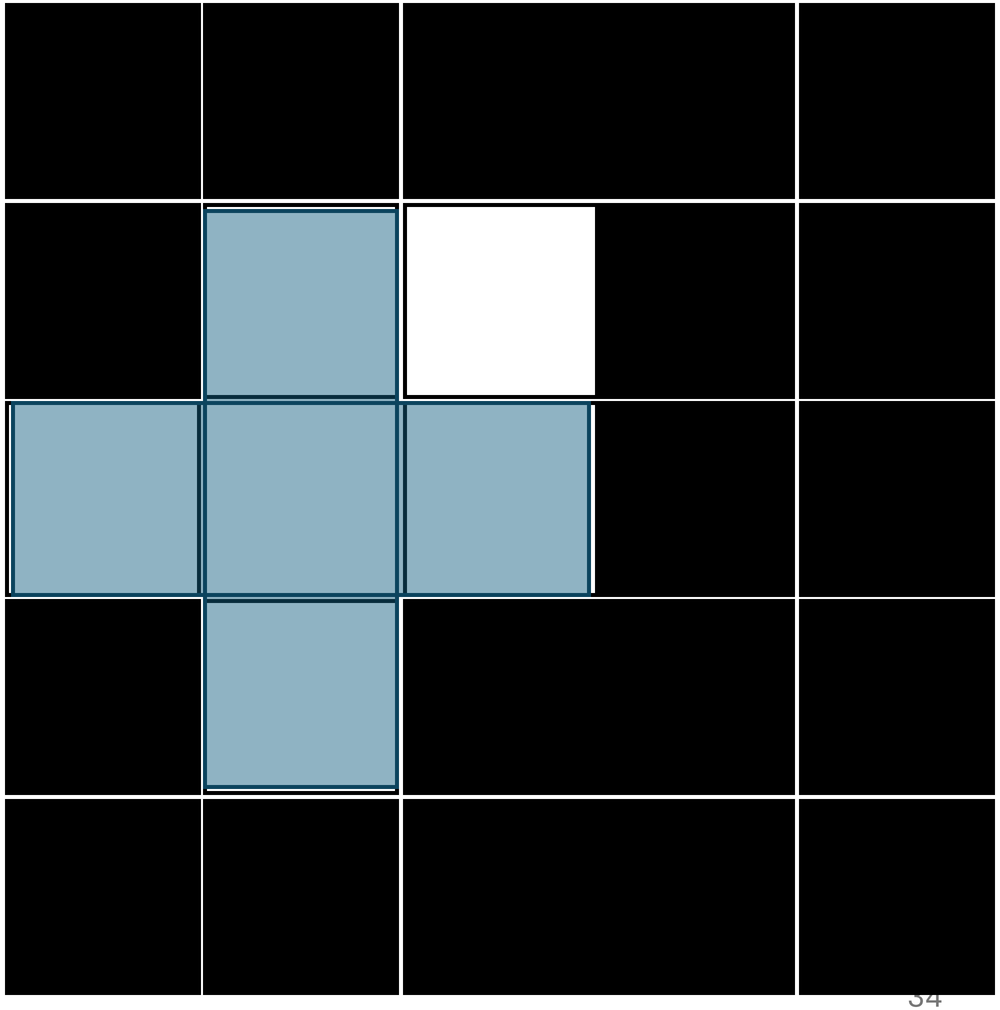
Morphological operat**ors**

- **Structuring Element** defines neighborhood around input pixel
- Set output pixel to „1“ if
  - all neighborpixels are „1“ (**Erosion**)
  - at least one of them is „1“ (**Dilation**)

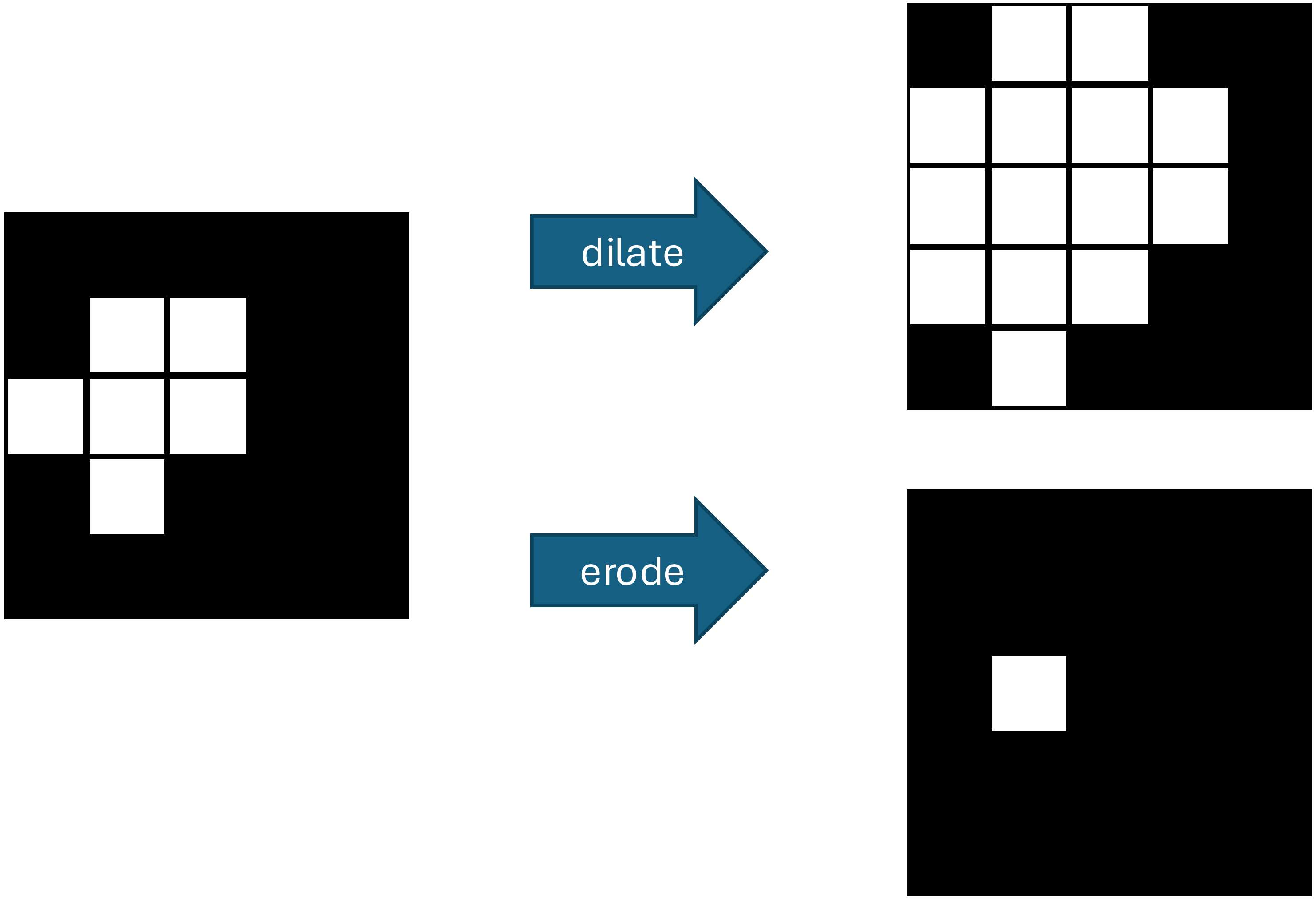


Morphological operators

- **Structuring Element** defines neighborhood around input pixel
- Set output pixel to „1“ if
  - all neighborpixels are „1“ (**Erosion**)
  - at least one of them is „1“ (**Dilation**)

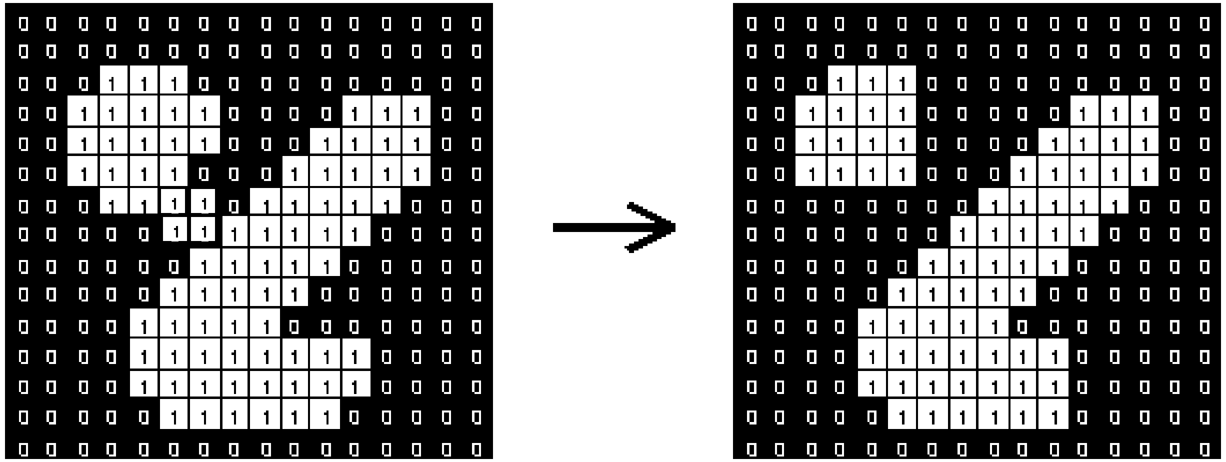


Morphological operators

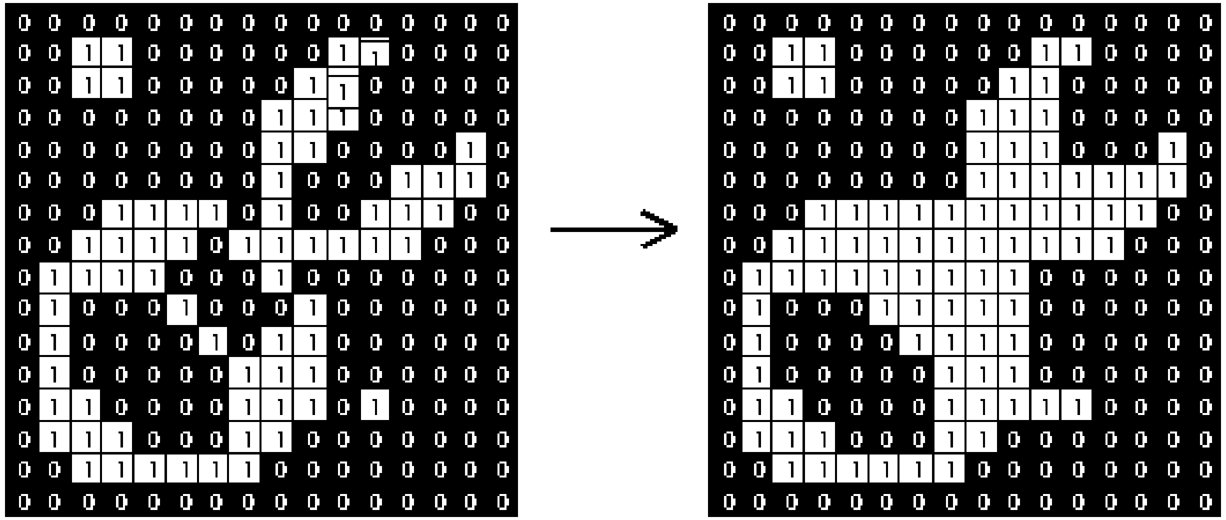


Morphological Operations

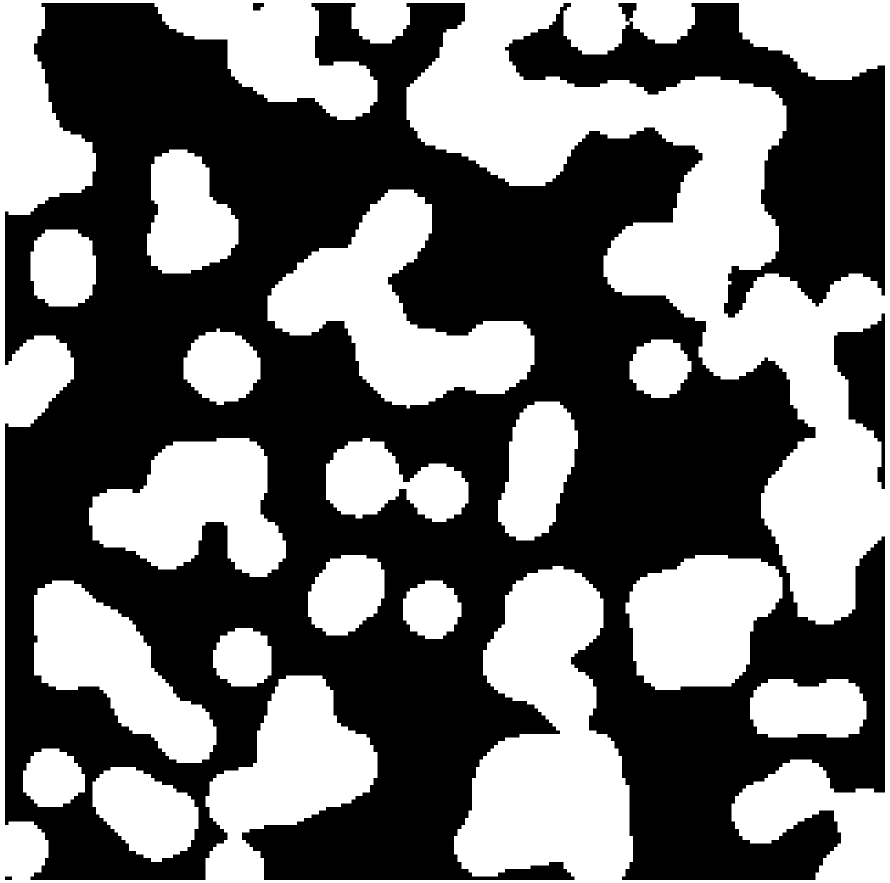
Opening(I) = Dilate(Erode(I))

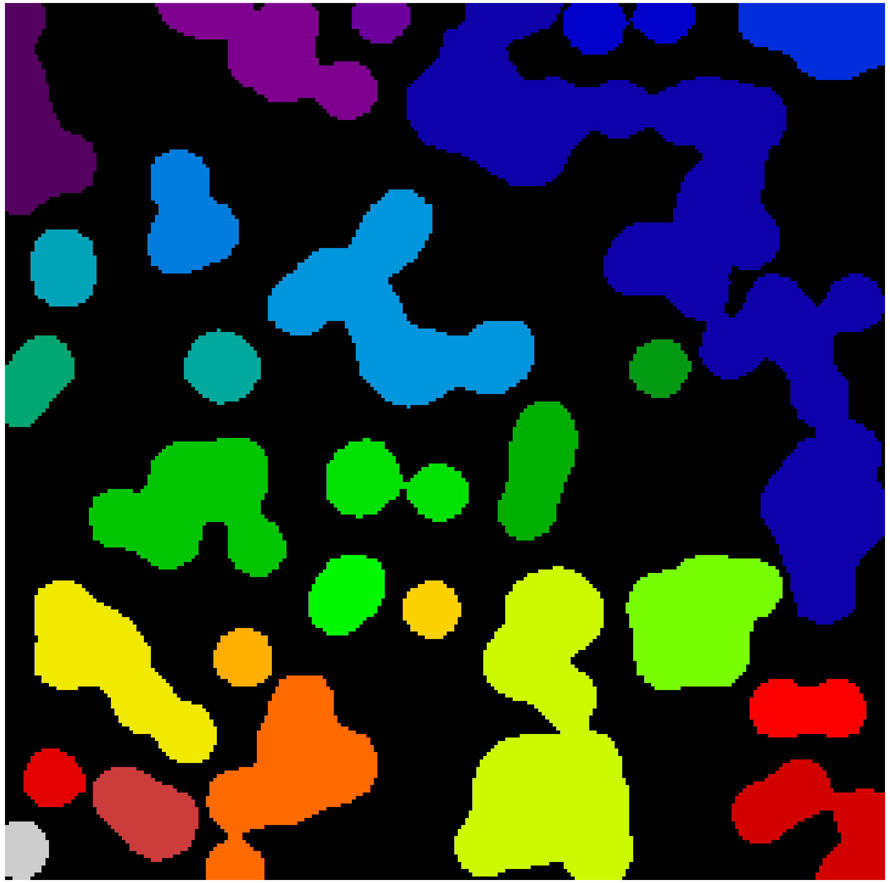


Closing(I) = Erode(Dilate(I))



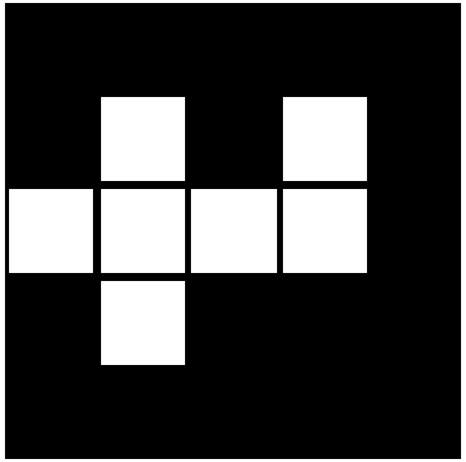
Find connected components

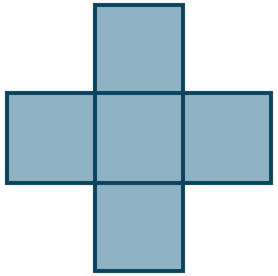




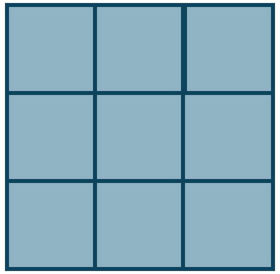
https://www.scipy-lectures.org/packages/scikit-image/auto_examples/plot_labels.html

Find connected components





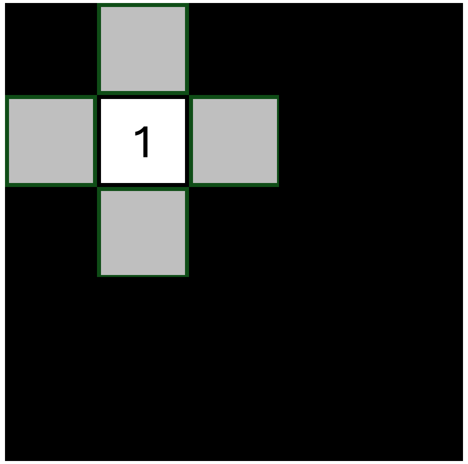
4-neighborhood



8-neighborhood

Today: The TWO-PASS Algorithm!

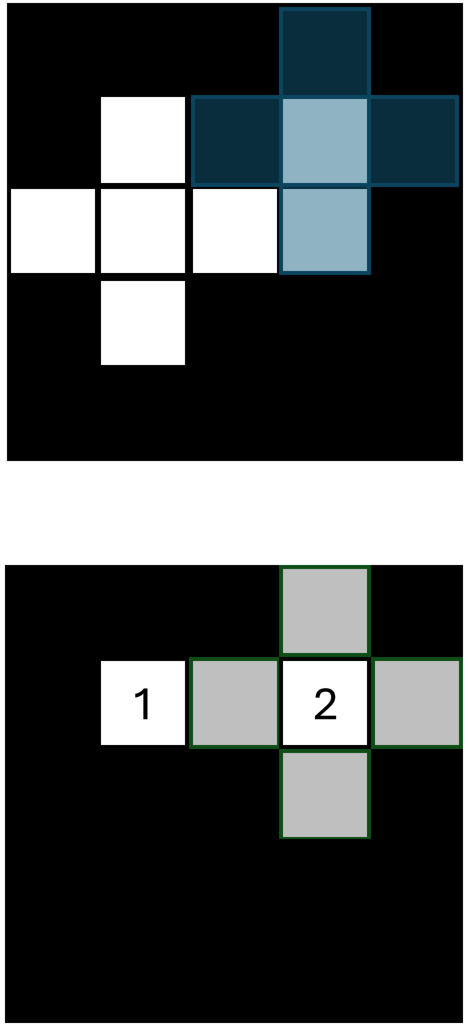
Find connected components



FIRST PASS!

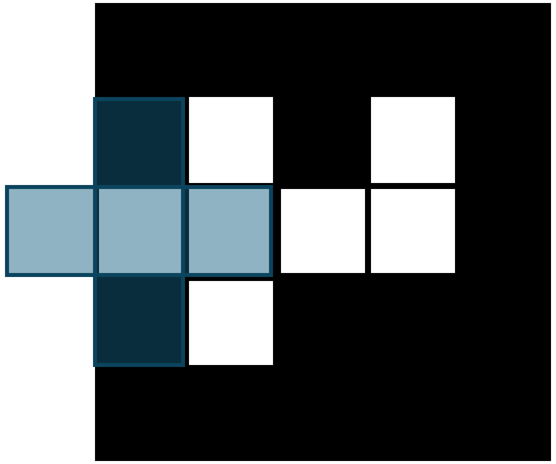
- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If not: assign new label

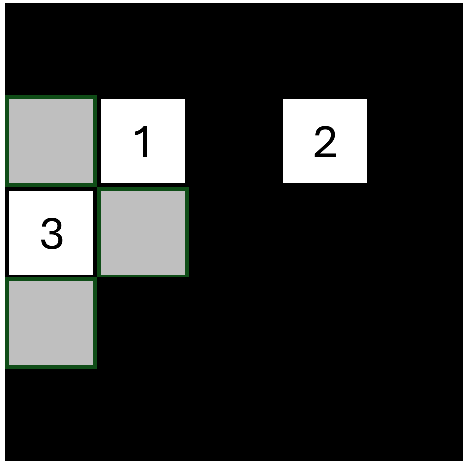
Find connected components



- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If not: assign new label

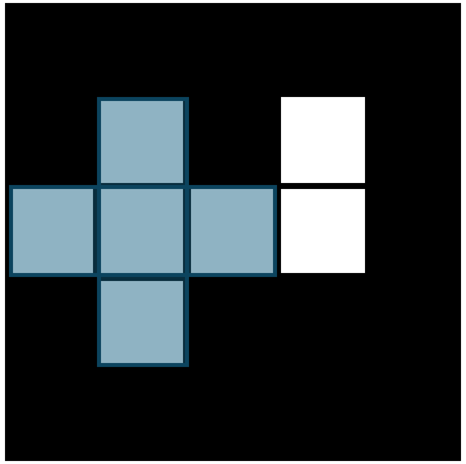
Find connected components

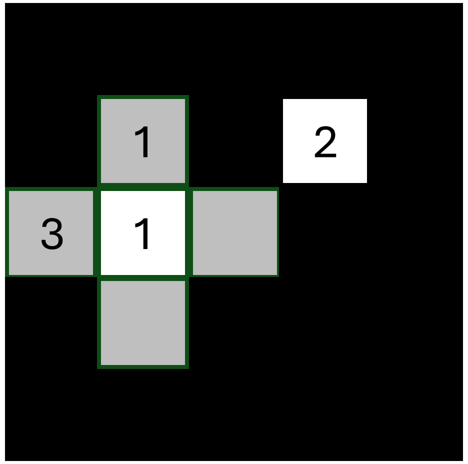




- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If not: assign new label

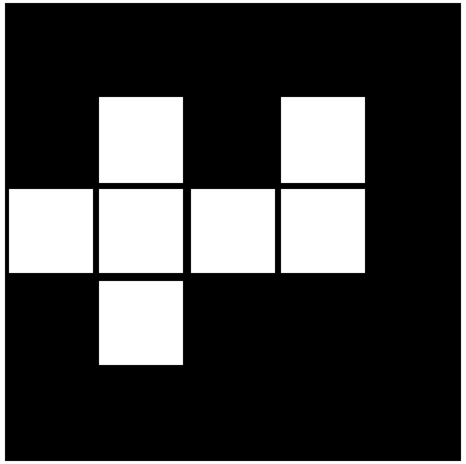
Find connected components

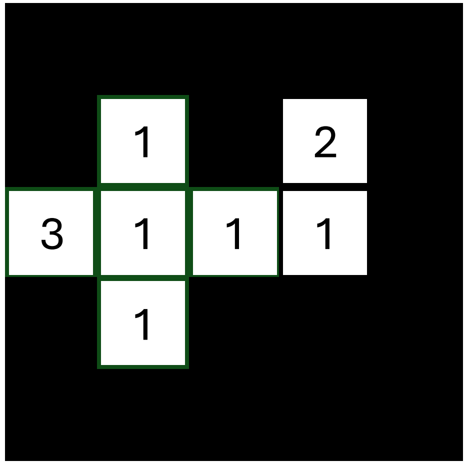




- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If not: assign new label
    - If yes: assign minimum

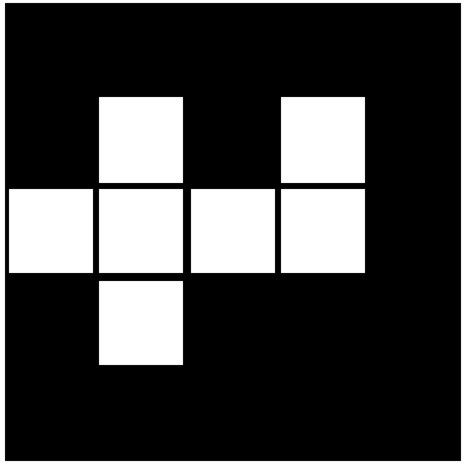
Find connected components

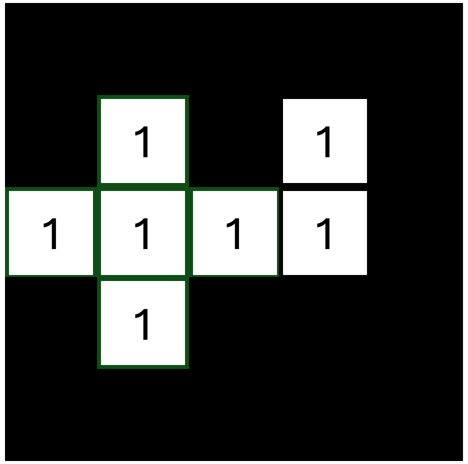




- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If not: assign new label
    - If yes: assign minimum

Find connected components





SECOND PASS!

- Scan the input image, pixel by pixel
- If the center pixel is 1
  - check if neighbors have been labeled
    - If yes: assign minimum

Ex. 1.2: connected components in color detections

- starting from the binary color detection image
- erase noise with an erosion operation
- dilate once to get original size of object
- find connected components with two-pass algorithm
- challenge 1: can you do it with one pass?
- challenge 2: extract bounding box „on-the-fly“
  - draw bounding box on original image

Another popular feature: brightness differences to background

- Static scene with (more/less) static background?
- Objects discriminable from background?
- use “Background Subtraction”!

Background Subtraction

- estimate a **background model** (sometimes „**background hypothesis**“)
- the simplest variant is integrating all frames into one background image:

$$B_t = \alpha B_{t-1} + (1-\alpha) I_t$$
$$0 < \alpha < 1$$

- This is called an **exponential average:** the influence of an image exponentially decays over time

Background Subtraction

- Difference to background model: $D_t = |B_t - I_t|$
- Difference larger than threshold? $D_t > \theta\ ?\ 1 : 0$
- Post-process binary image with
  - Erosion / Dilation
  - Connected components analysis