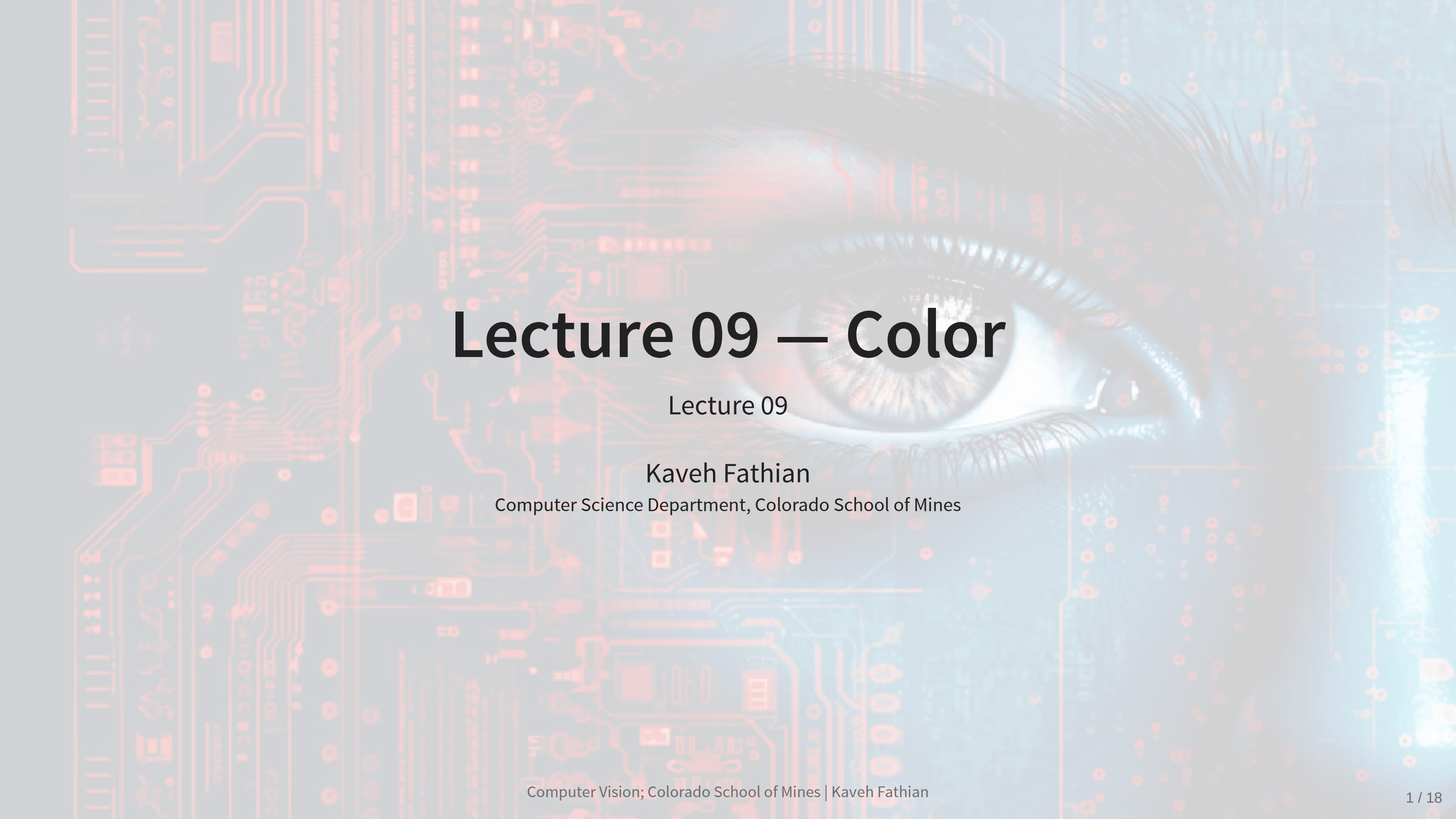


Lecture 09 — Color

Lecture 09

Kaveh Fathian
Computer Science Department, Colorado School of Mines

Learning Objectives

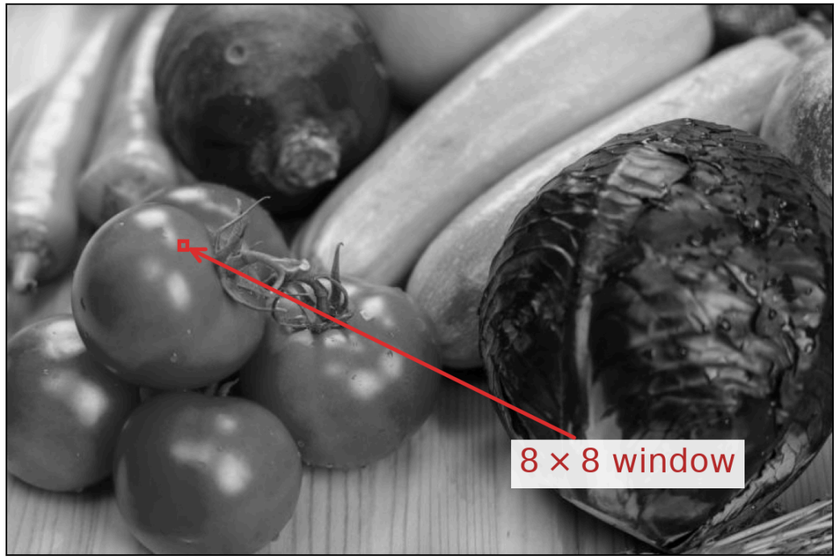
By the end of this lecture, you should be able to:

- Explain how humans perceive color and how cameras capture it.
- Represent color images as arrays and access their channels in OpenCV.
- Compare common color spaces and describe their uses.
- Use HSV thresholds to segment and track a colored object.

Grayscale Images Are Matrices

- A grayscale image is an $H \times W$ matrix: one intensity value per pixel.
- For an 8-bit image, values range from **0 (black)** to **255 (white)**.

Grayscale image: 512 rows × 768 columns



Pixel values in the window

Row index y \ Column index x	160	161	162	163	164	165	166	167
220	140	136	131	128	128	130	132	132
221	140	136	131	128	128	130	132	133
222	139	136	132	129	129	131	132	132
223	139	136	132	130	130	131	132	132
224	138	134	129	127	128	129	129	128
225	136	133	128	127	127	128	128	128
226	132	130	127	125	126	127	127	127
227	128	126	125	124	124	125	126	126

- Python indexes the image as `gray[y, x]`: **row first, column second**.
- The top-left pixel is `gray[0, 0]`.

RGB Images Have Three Channels

- An RGB image contains three $H \times W$ matrices: **red**, **green**, and **blue**.
- Together, they form an array with shape $(H, W, 3)$.

$$I(y, x) = [\, R(y, x) \quad G(y, x) \quad B(y, x) \,]$$

RGB image



Red contribution



Green contribution



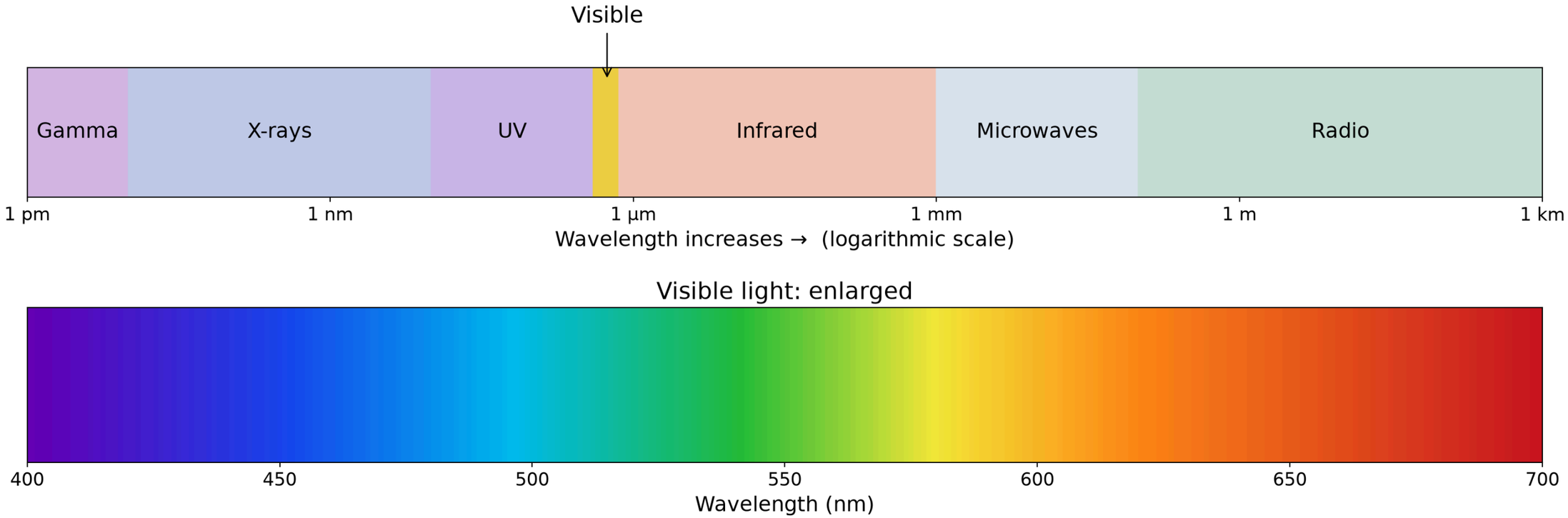
Blue contribution



- Each pixel has three values. For example, $(255, 0, 0)$ is red and $(255, 255, 255)$ is white.

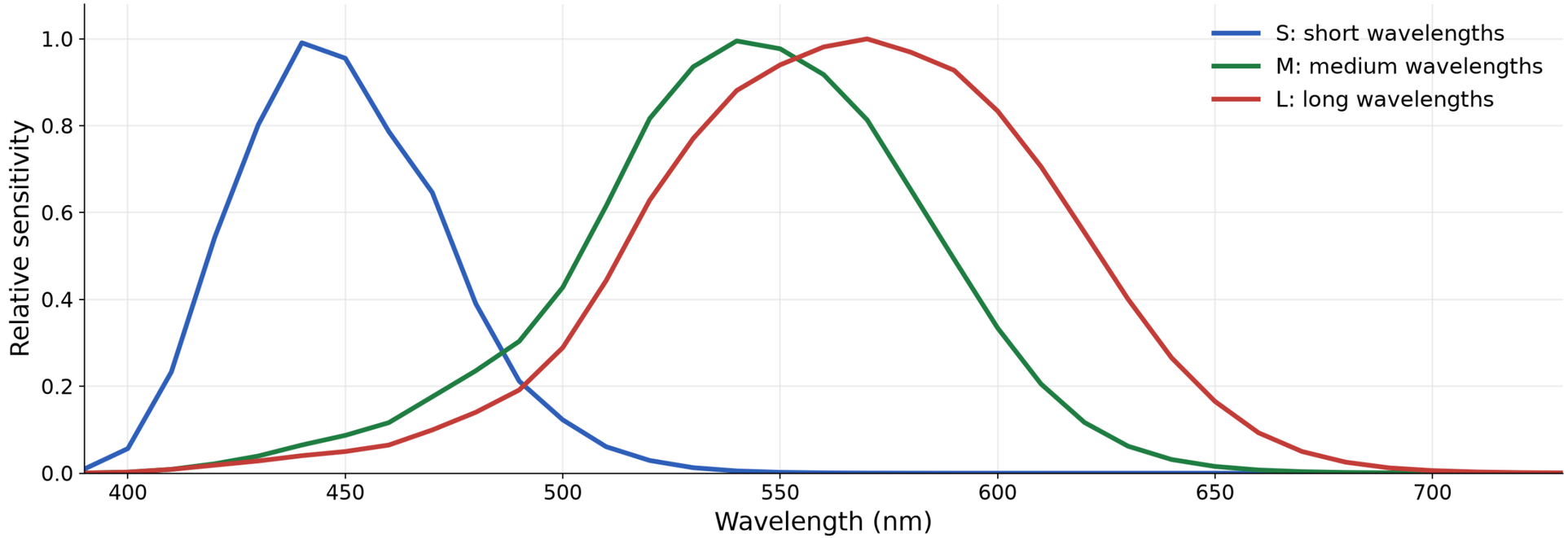
Light and Wavelength

- Visible light is a small part of the electromagnetic spectrum, approximately **400–700 nm**.
- A single wavelength has a spectral color. Most light reaching a camera contains a mixture of wavelengths.



Human Color Perception

- In daylight, color vision relies on three types of cone cells: **S**, **M**, and **L**.
- Their sensitivities overlap. Color depends on the combination of their responses.

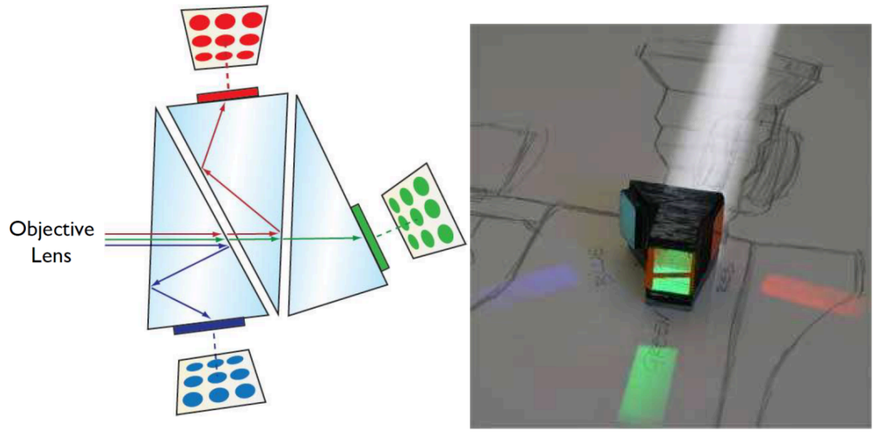


- S, M, and L refer to wavelength sensitivity; they are not the RGB channels of an image.

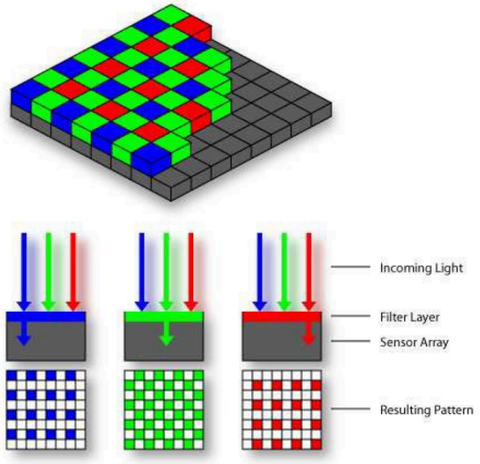
How Cameras Capture Color

- **Three sensors:** a prism separates light into three paths, one for each color channel.
- **One sensor:** a color filter array lets each sensor location measure one color component.

Three-sensor camera



Filter array over one sensor



Bayer pattern (RGGB)

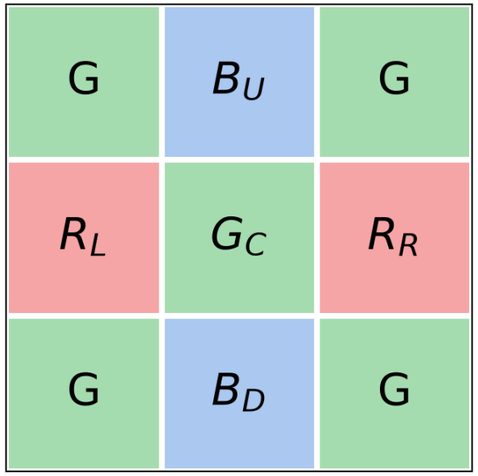
R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B

- Each 2×2 Bayer block contains **two green**, **one red**, and **one blue** sample.
- Human vision is most sensitive to green, where the M and L cones overlap, so green carries most of perception.
- The missing color components must be estimated to obtain an RGB image.

Reconstructing RGB: Demosaicing

Demosaicing estimates the two missing color components at each sensor location using nearby samples.

A green location and its neighbors



At the center, green is measured directly:

$$\widehat{G}_C = G_C$$

Estimate red and blue by averaging nearby samples:

$$\widehat{R}_C = \frac{R_L + R_R}{2}, \qquad \widehat{B}_C = \frac{B_U + B_D}{2}$$

The reconstructed pixel is

$$\hat{I}_C = \begin{bmatrix} \widehat{R}_C & G_C & \widehat{B}_C \end{bmatrix}.$$

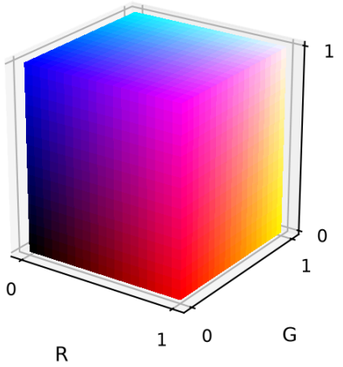
- This is a simple **bilinear interpolation** example. The neighbors used depend on the location's filter color and position in the Bayer pattern.

Color Spaces: RGB

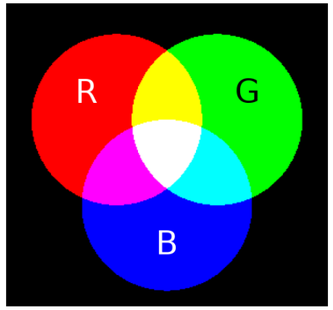
- A color space represents colors numerically. RGB uses **red, green, and blue** components and is widely used for images and displays.
- RGB is an **additive** model: adding colored light produces brighter colors.

```
1 bgr = cv2.imread("assets/09_color/rgb.jpg")
2 rgb = cv2.cvtColor(bgr, cv2.COLOR_BGR2RGB)
3 red, green, blue = cv2.split(rgb)
```

RGB cube (0–1)



Adding light



Red channel



Green channel



Blue channel

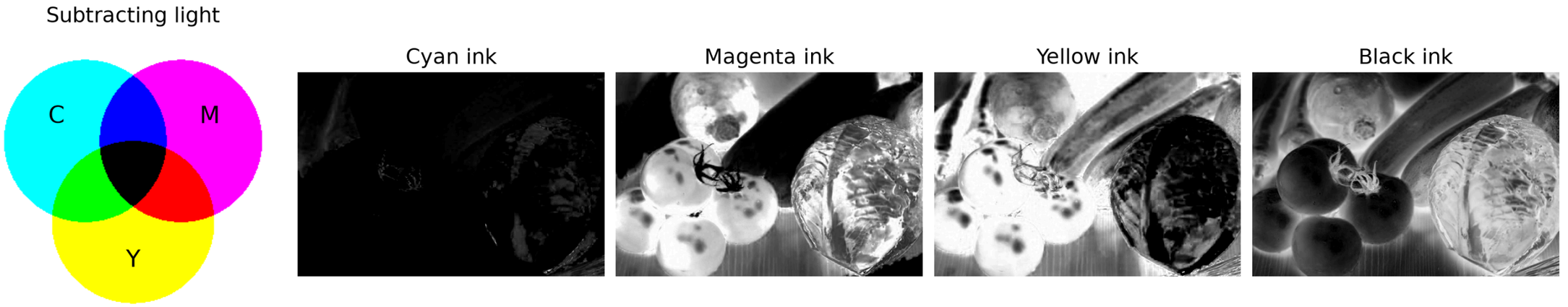


OpenCV loads **BGR**, so we convert to RGB before splitting. In the channel images, brighter pixels mean larger values (0–255).

CMYK: Color for Printing

- CMYK uses cyan, magenta, yellow, and black inks in a **subtractive** model: inks absorb light.
- Used for printing. The code below illustrates a simplified conversion; actual printing uses color profiles.

```
rgb_float = rgb.astype(np.float32) / 255
black = 1 - rgb_float.max(axis=2)
cmy = (1 - rgb_float - black[..., None]) / np.maximum(1 - black[..., None], 1e-6)
cyan, magenta, yellow = cv2.split(cmy)
```

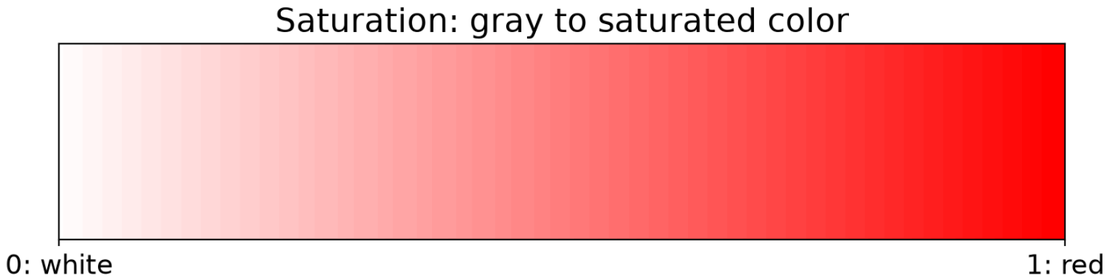


Channel values range from 0 to 1; brighter pixels indicate more ink. OpenCV has no direct CMYK conversion, so this example uses NumPy.

HSV: Hue, Saturation, and Value

- **Hue** describes the color, **saturation** its departure from gray, and **value** the largest RGB component.
- HSV is useful for selecting and tracking objects by color.



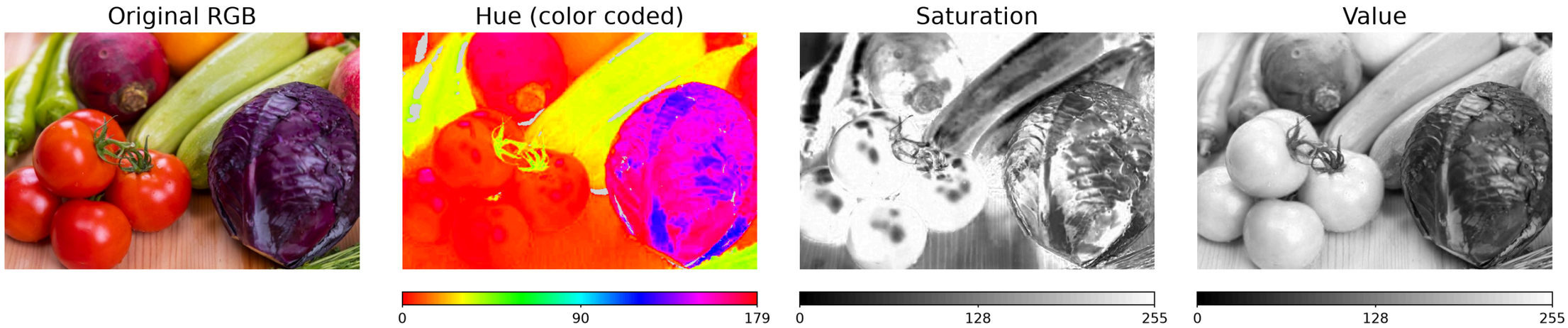




HSV in Python

- Hue separates colors; saturation and value help exclude nearly gray or very dark pixels.
- Hue is unreliable when saturation or value is very small.

```
hsv = cv2.cvtColor(bgr, cv2.COLOR_BGR2HSV)
hue, saturation, value = cv2.split(hsv)
```

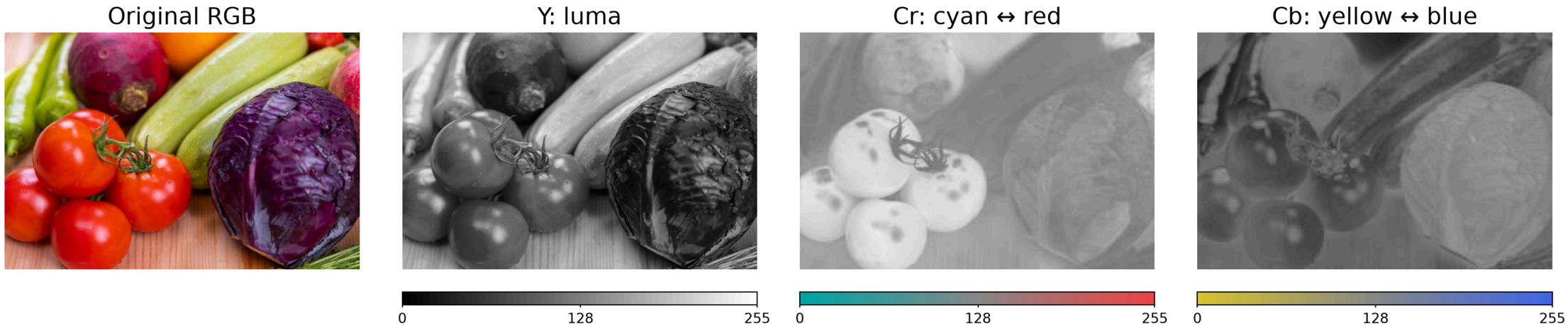


For 8-bit OpenCV images: $H \in [0, 179]$ (approximately half the angle in degrees), and $S, V \in [0, 255]$.

YCbCr: Luma and Chroma

- **Y** represents luma; **Cb** and **Cr** represent blue- and red-difference chroma.
- Used in image/video compression: chroma can be stored at lower resolution than luma.

```
1 ycrcb = cv2.cvtColor(bgr, cv2.COLOR_BGR2YCrCb)
2 luma, cr, cb = cv2.split(ycrcb)
```

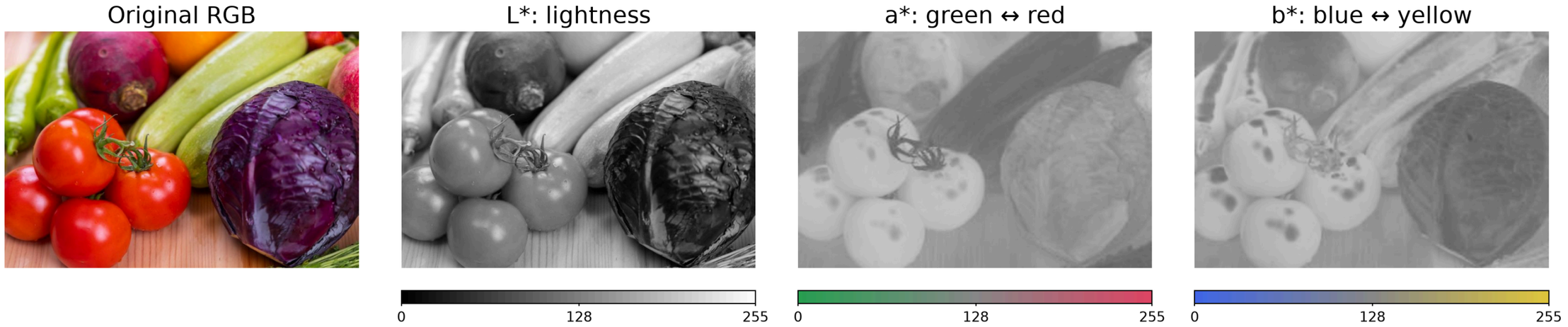


OpenCV returns channels in **Y, Cr, Cb** order. For 8-bit images, neutral chroma is approximately 128.

CIELAB: Lightness and Color Differences

- L^* describes lightness; a^* runs from green to red, and b^* from blue to yellow.
- Approximately perceptually uniform: distances are useful for comparing colors and grouping similar colors.

```
lab = cv2.cvtColor(bgr, cv2.COLOR_BGR2Lab)
lightness, a_channel, b_channel = cv2.split(lab)
```



OpenCV's 8-bit encoding: $L_8 = 255L^*/100, a_8 = a^* + 128, b_8 = b^* + 128$. Neutral $a^* = b^* = 0$ is stored as 128.

Color-Based Segmentation

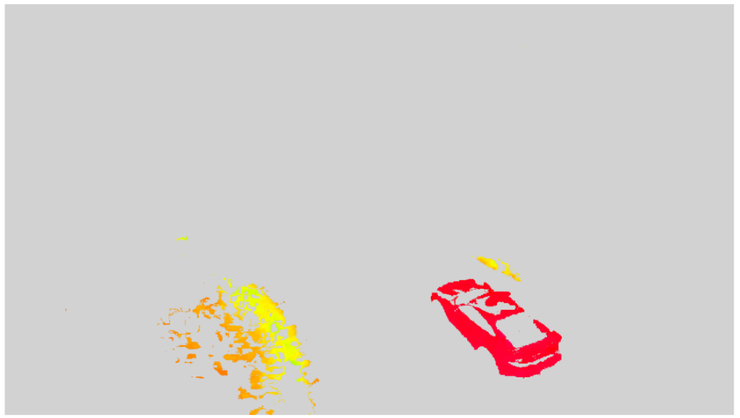
- Select red pixels in HSV. Red wraps around the hue scale, so we combine **two hue ranges**.
- Minimum saturation and value thresholds reject weak colors and dark pixels.

```
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv, (0, 130, 80), (5, 255, 255)) # returns 255 for matching pixels, 0 othe
mask |= cv2.inRange(hsv, (170, 130, 80), (179, 255, 255)) # |= combines the masks
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, np.ones((3, 3), np.uint8)) # fills small gaps
```

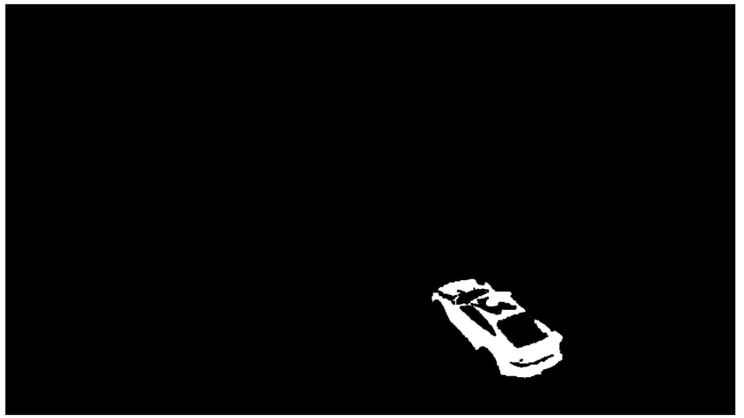
Input frame



Hue (S ≥ 130 and V ≥ 80)



Binary mask



Tracking the Red Car: Complete Code

Select the largest red region in each frame, draw its bounding box, and retain its recent center positions.

```python
import cv2
import numpy as np
from collections import deque

def annotate_car(frame, trail):
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    mask = cv2.inRange(hsv, (0, 130, 80), (5, 255, 255))
    mask |= cv2.inRange(hsv, (170, 130, 80), (179, 255, 255))
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, np.ones((3, 3), np.uint8))
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    result = frame.copy()
    if contours:
        car = max(contours, key=cv2.contourArea)
        if cv2.contourArea(car) > 10:
            x, y, w, h = cv2.boundingRect(car)
            trail.append((x + w//2, y + h//2))
            cv2.rectangle(result, (x, y), (x+w, y+h), (0, 255, 255), 3)
            cv2.circle(result, trail[-1], 4, (0, 255, 255), -1)

    if len(trail) > 1:
        cv2.polylines(result, [np.array(trail, np.int32)], False, (255, 255, 0), 2)
```

Color-Based Tracking: Result

- Yellow box: detected red region.
- Cyan line: the last 40 detected centers.
- The trail is in image coordinates, so it also reflects camera motion.



Key Takeaways

- An RGB image contains three channels. OpenCV loads color images in BGR order.
- Human color vision relies on three cone types. Bayer cameras measure one color component per sensor location and estimate the missing components through demosaicing.
- Color spaces emphasize different properties: RGB for display, HSV for color selection, YCbCr for compression, and Lab for perceptual color comparison.
- Color segmentation combines hue, saturation, and value thresholds. Hue alone may not distinguish an object from its background.