Robot Vision Basics

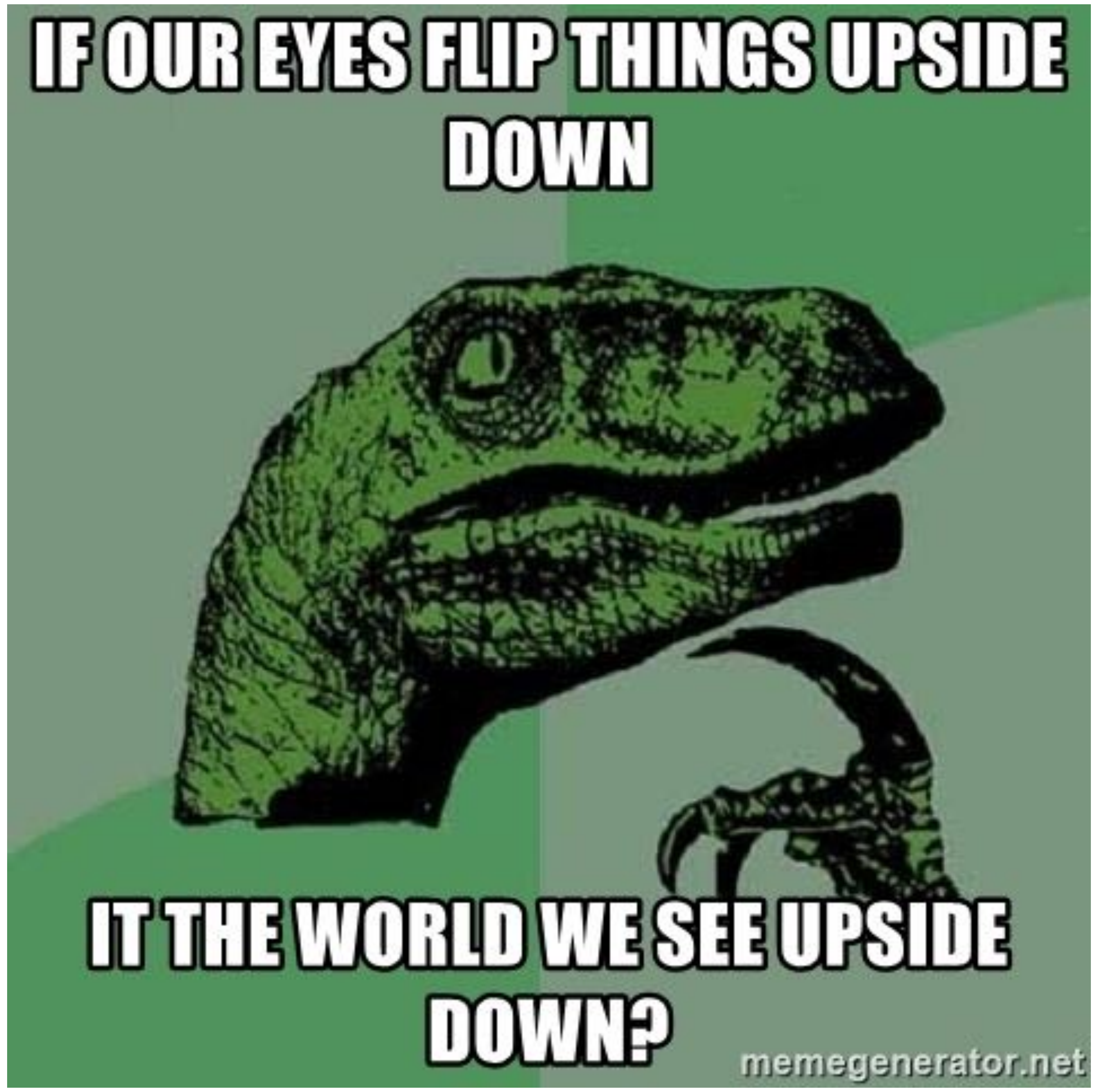
Outline

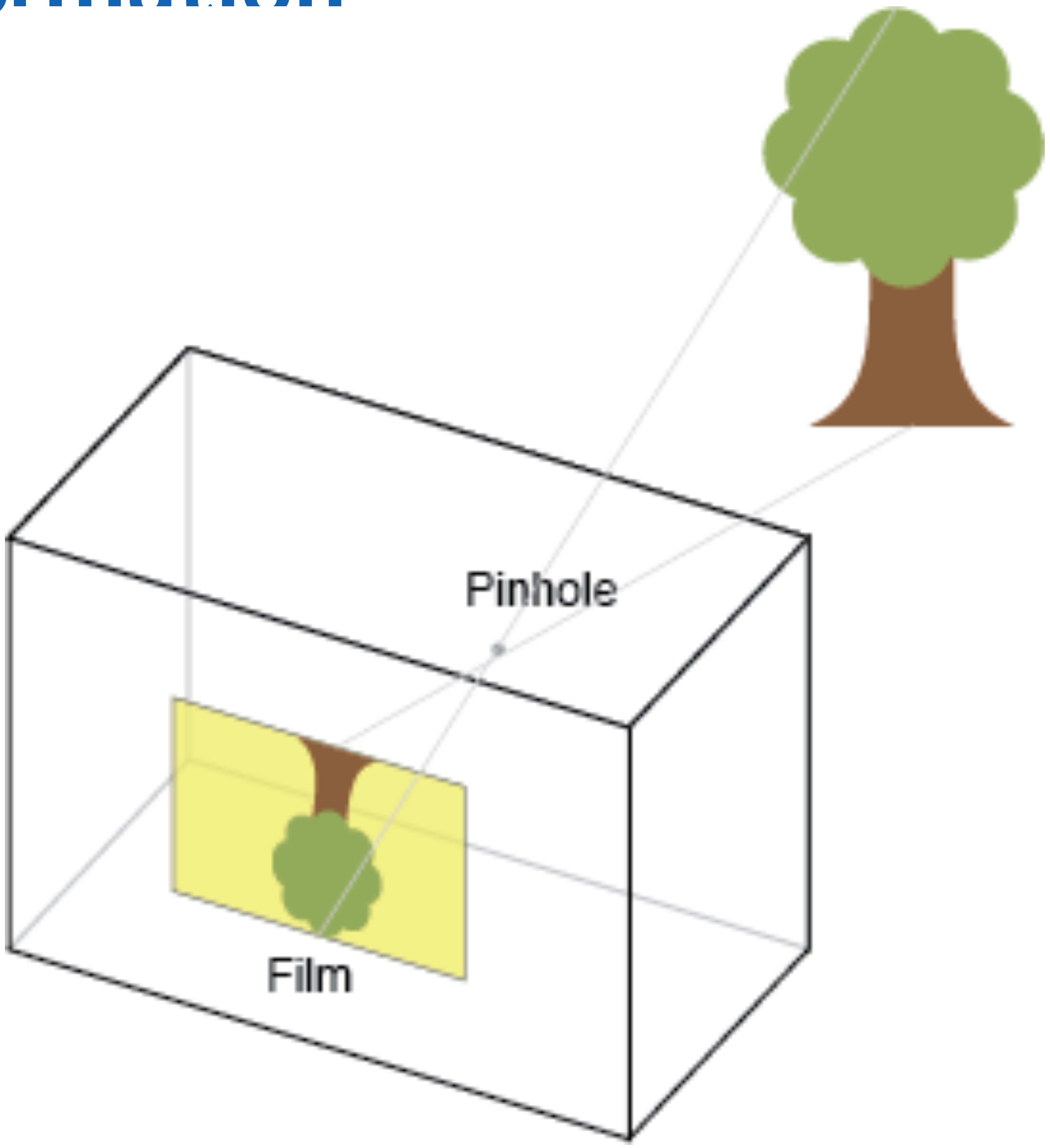
- ~~Image formation basics~~
    - ~~How images are represented~~
    - ~~Basic image manipulation, transformations~~
    - Detections and classification —> mini-project
- Camera operations
    - Camera properties, projections
    - Calibration, pose estimation, fun AR exercise

Image Formation Basics

Image Formation

- Pin-hole camera model — images are turned upside down!

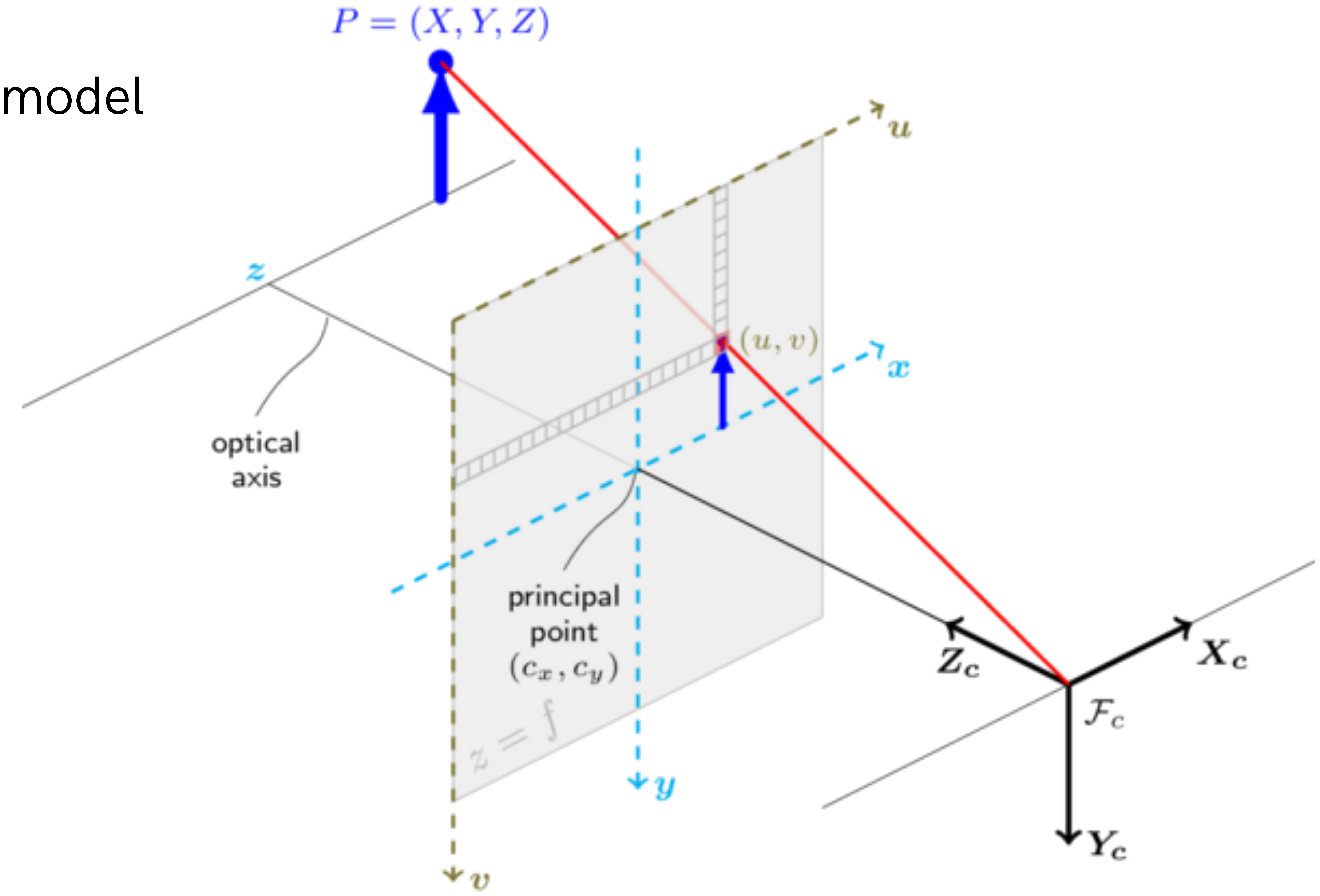




[http://ksimek.github.io/pinhole_camera_diagram/]

Image Formation

- Pin-hole camera model
- Pixels
- Image axes

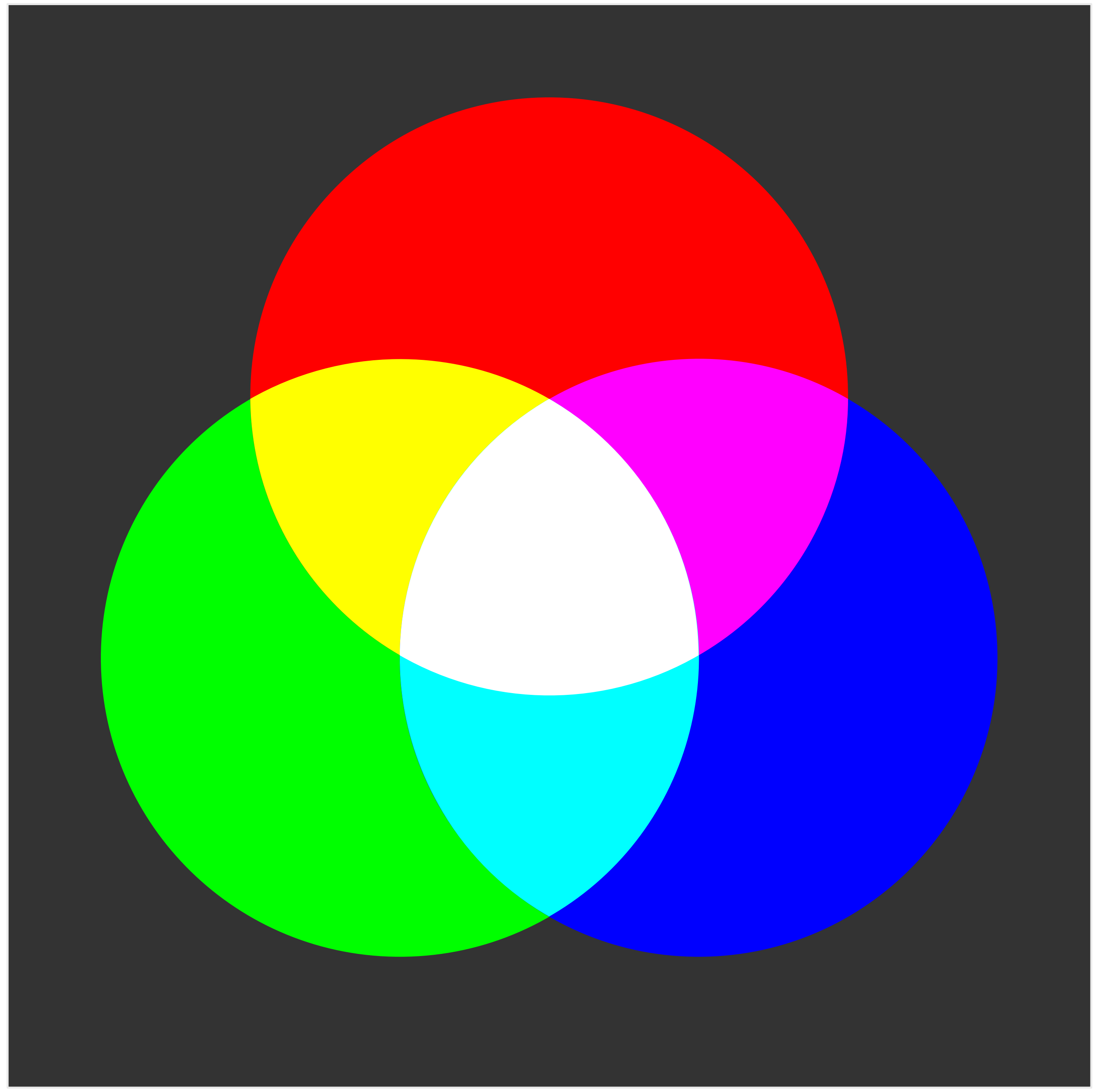


[docs.opencv.org]

Image Base Colors

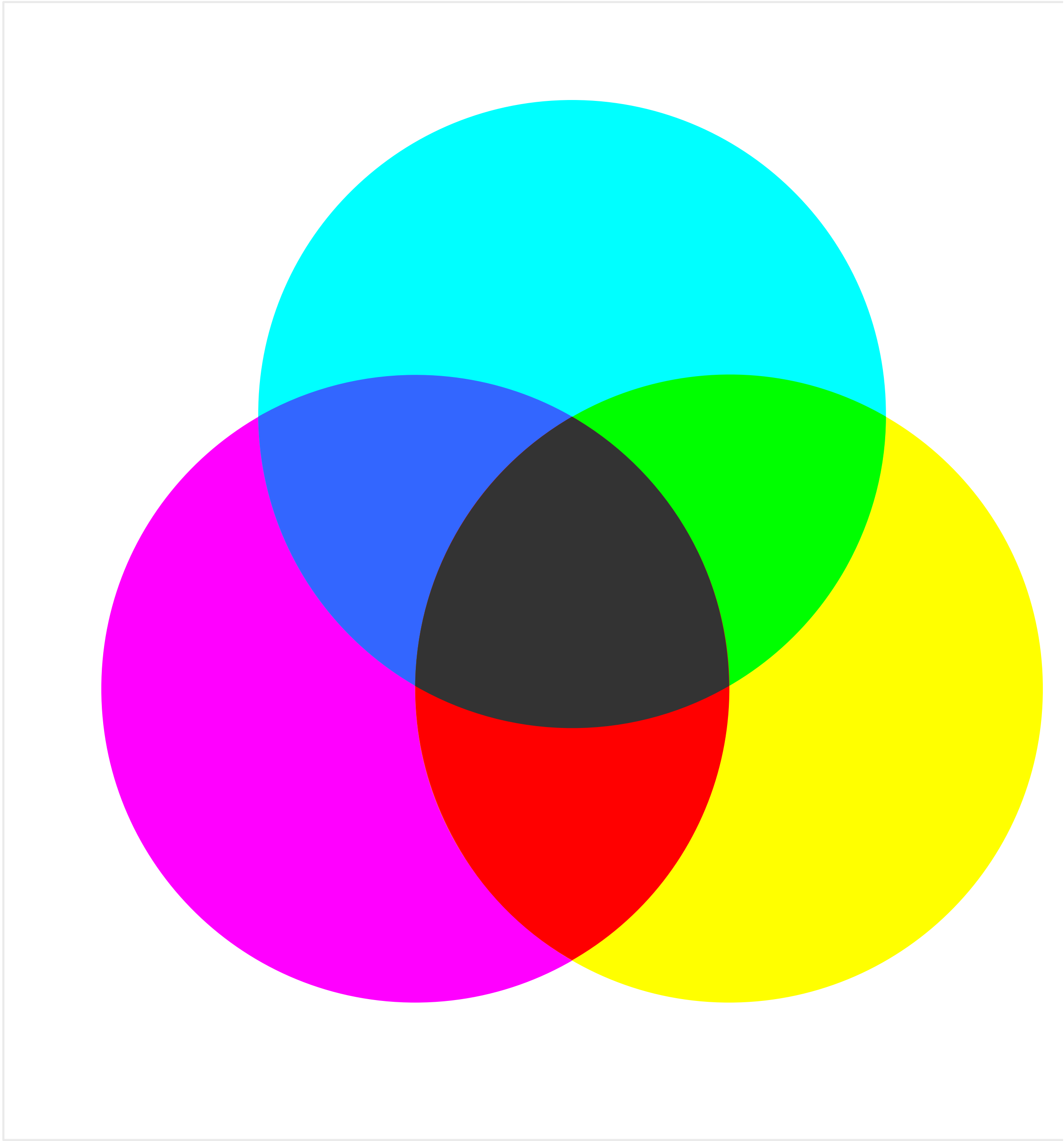
- Primary and secondary colors
  - Each set can be mixed to produce the other set.

NOTE: opencv uses a BGR convention



(a)

RGB



(b)

CMY

[Szeliski ’10]

Image Representation

- Images are represented using matrices.
- Each primary color is a separate matrix, so that a full color image has three matrices. These are sometimes called **channels**.
- Each cell in the matrix is a pixel, and contains an integer value between 0 and 255
- Example: Black (0, 0, 0), Yellow (0, 255, 255)
- Image coordinates usually have Y downwards

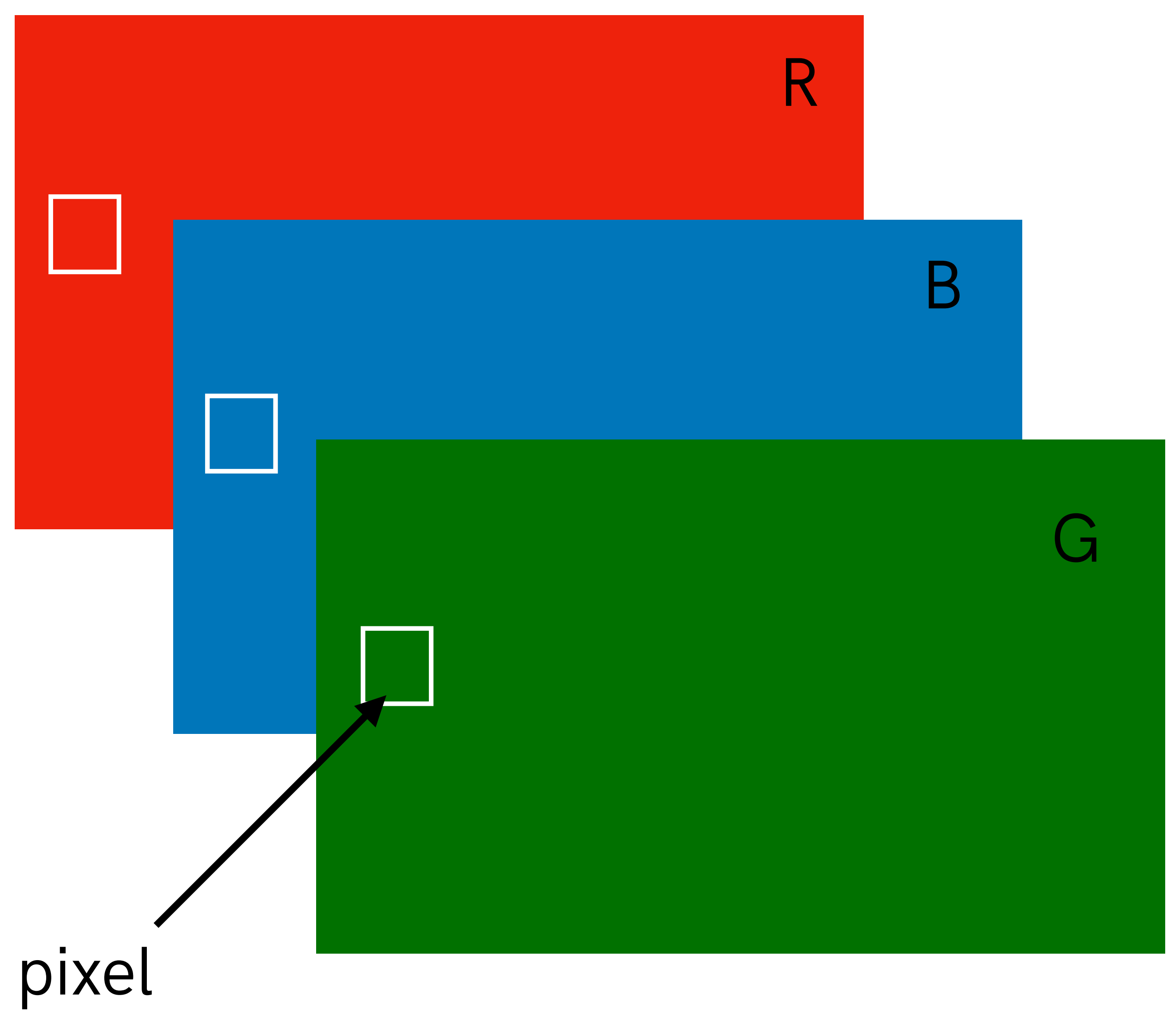


Image Manipulation

- Since an image is represented by a matrix;
- We use basic matrix transformation in 2D
- Given an image of size m x n and having c channels, $I_{m,n,c}$
- Update/manipulate range of pixels:

```
image[start:end, start:end, channel] = new_value      # Update pixels in one channel
image[start:end, start:end, :] = new_value            # Update pixels in all channel
```

- Separate into individual channels:

```
blue_channel, green_channel, red_channel = cv2.split(image)
```


NOTE: opencv uses a BGR convention

opencv Basics

- Open source computer vision library
  - Written in C/C++ with bindings for Python — which we will use
- Has most common operations on images, video implemented.
- Has utilities for interacting with cameras

```
import cv2
from matplotlib import pyplot as plt
camera = cv2.VideoCapture(0)
ret, image  = camera.read()
plt.imshow(image[:, :, ::-1])
# additional manipulation
```

See notebook for practical details

Mini Project I: Image Transformer

- Implement an image transformer python module with *methods* the following transformations
  - Translation — `def translate(image, t_x, t_y):`
  - Scaling — `def scale(image, s_x, s_y):`
  - Rotation — `def rotate(image, c_x, c_y, angle):`
  - Translation + Rotation — `def translate_and_rotate(image, t_x, t_y, c_x, c_y, angle):`
  - Translation + Scaling — `def translate_and_scale(image, t_x, t_y, s_x, s_y):`

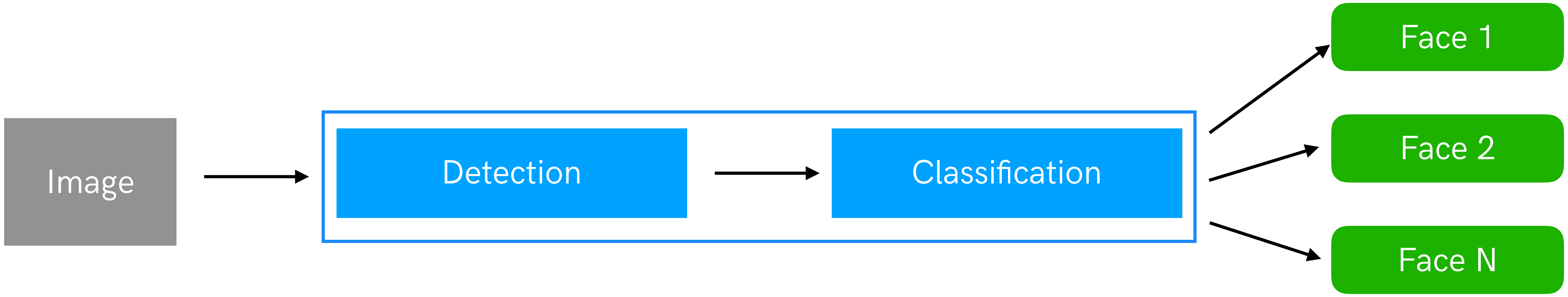
Video

Detection and Classification

- Given an image, we can search for, locate and intensify its semantic components

- For our application, it is important to detect faces so that we can anonymize them for privacy reasons.

- OpenCV provides a face detection API, which we will use.



opencv Face Detection Tutorial

Mini Project II: Privacy Enhancer

- Implement an image privacy enhancer python module that does the following

- Read in a stream of images (video) — use laptop camera
    - For each image:
        - Detect faces
        - Pixelate the face region so the identity is covered
        - Show the final image with pixelated face region

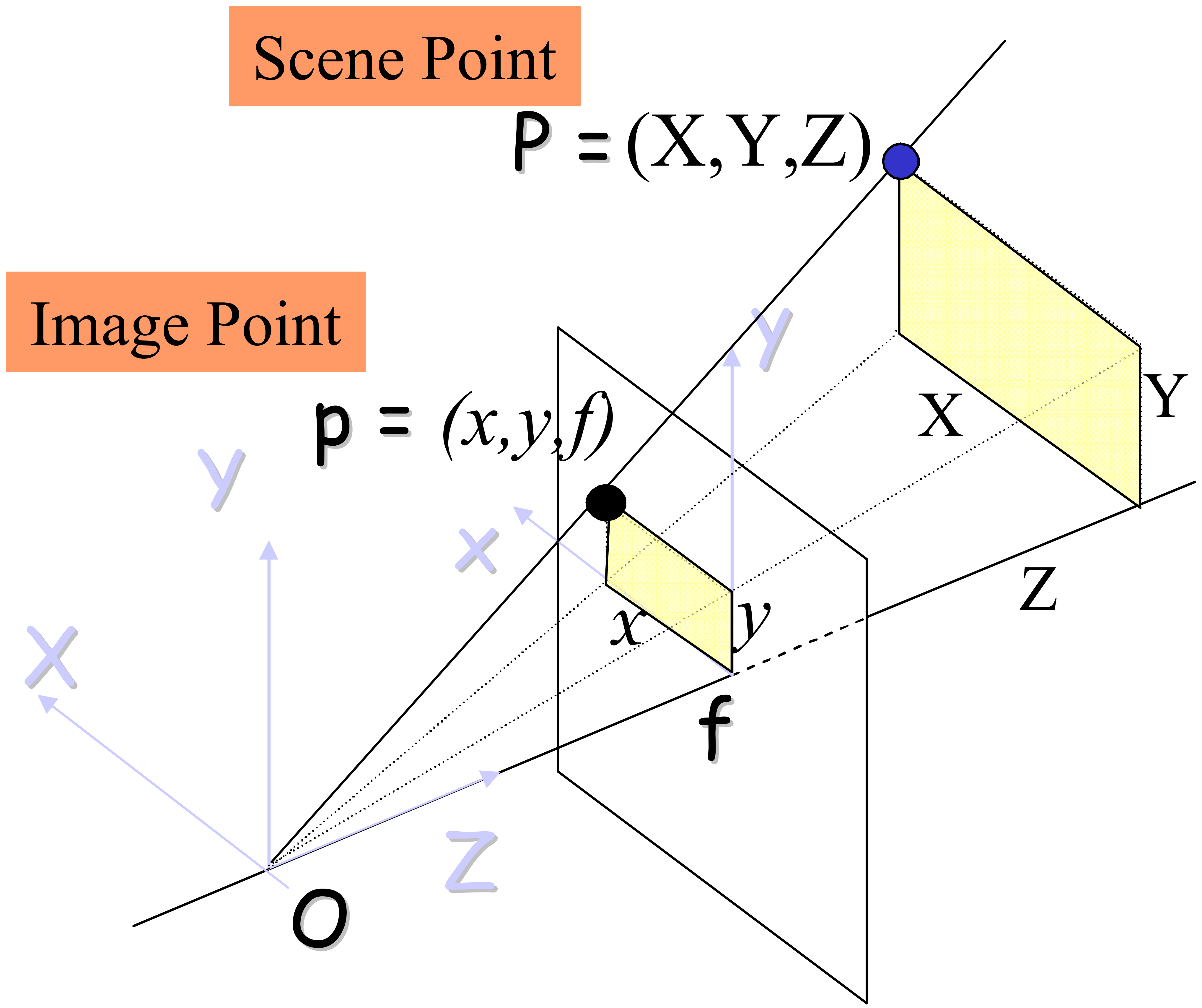
Free online image face pixelator [https://www.facepixelizer.com]

Camera Operations

Camera Projections

- **Perspective projection:** transformation of a point in the real world into a point on the camera image frame.

- **f** is part of camera parameters (focal length



Perspective Projection Eqns

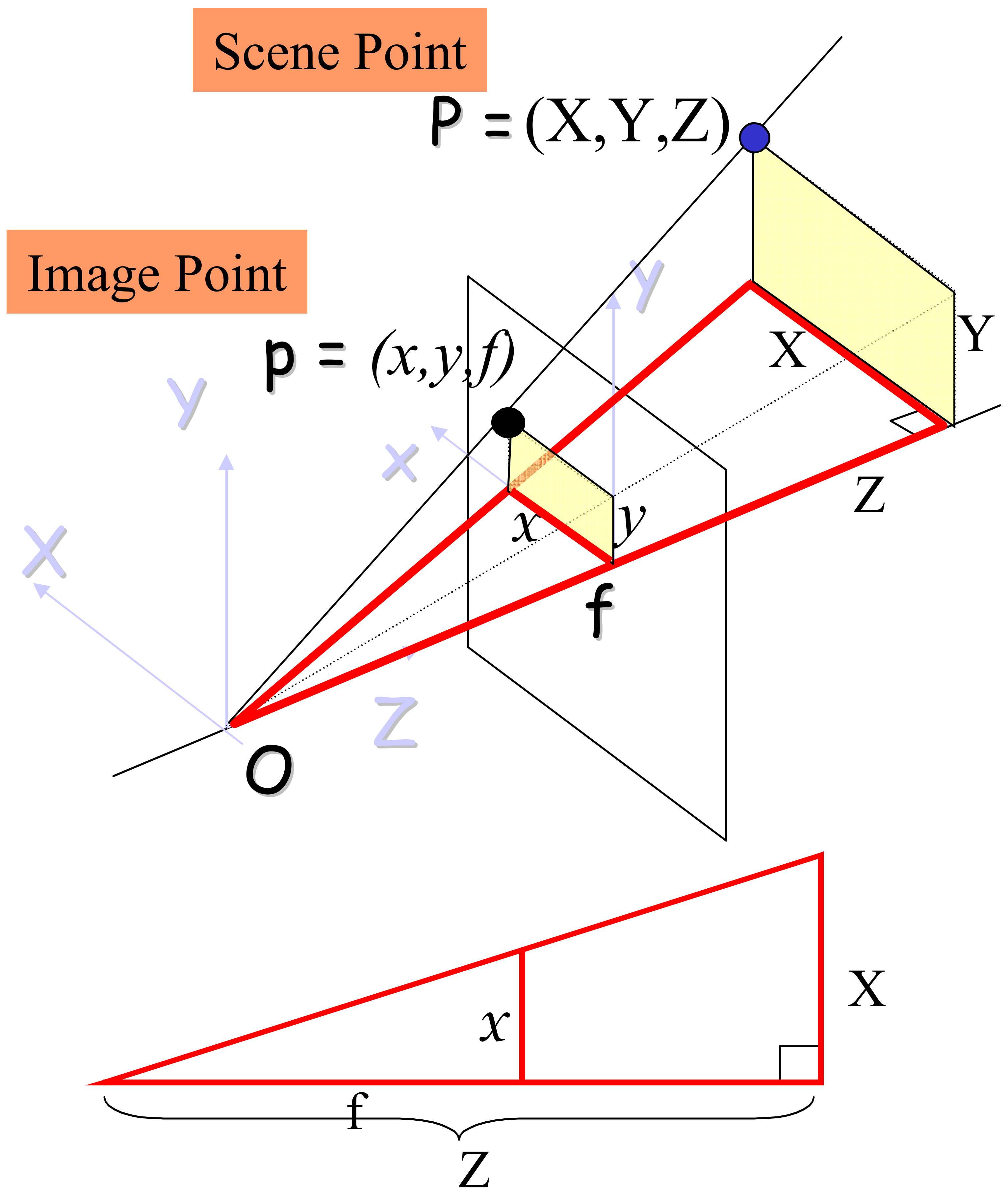
$$x = f\,\frac{X}{Z}$$

$$y = f\,\frac{Y}{Z}$$

Slide image from: Collins R. Penn State CSE486

Camera Projections

- **Perspective projection:** transformation of a point in the real world into a point on the camera image frame.
- x relations between scene and camera points.
- Z is the depth/distance of the point.



Perspective Projection Eqns

$$x = f\,\frac{X}{Z}$$

$$y = f\,\frac{Y}{Z}$$

Slide image from: Collins R. Penn State CSE486

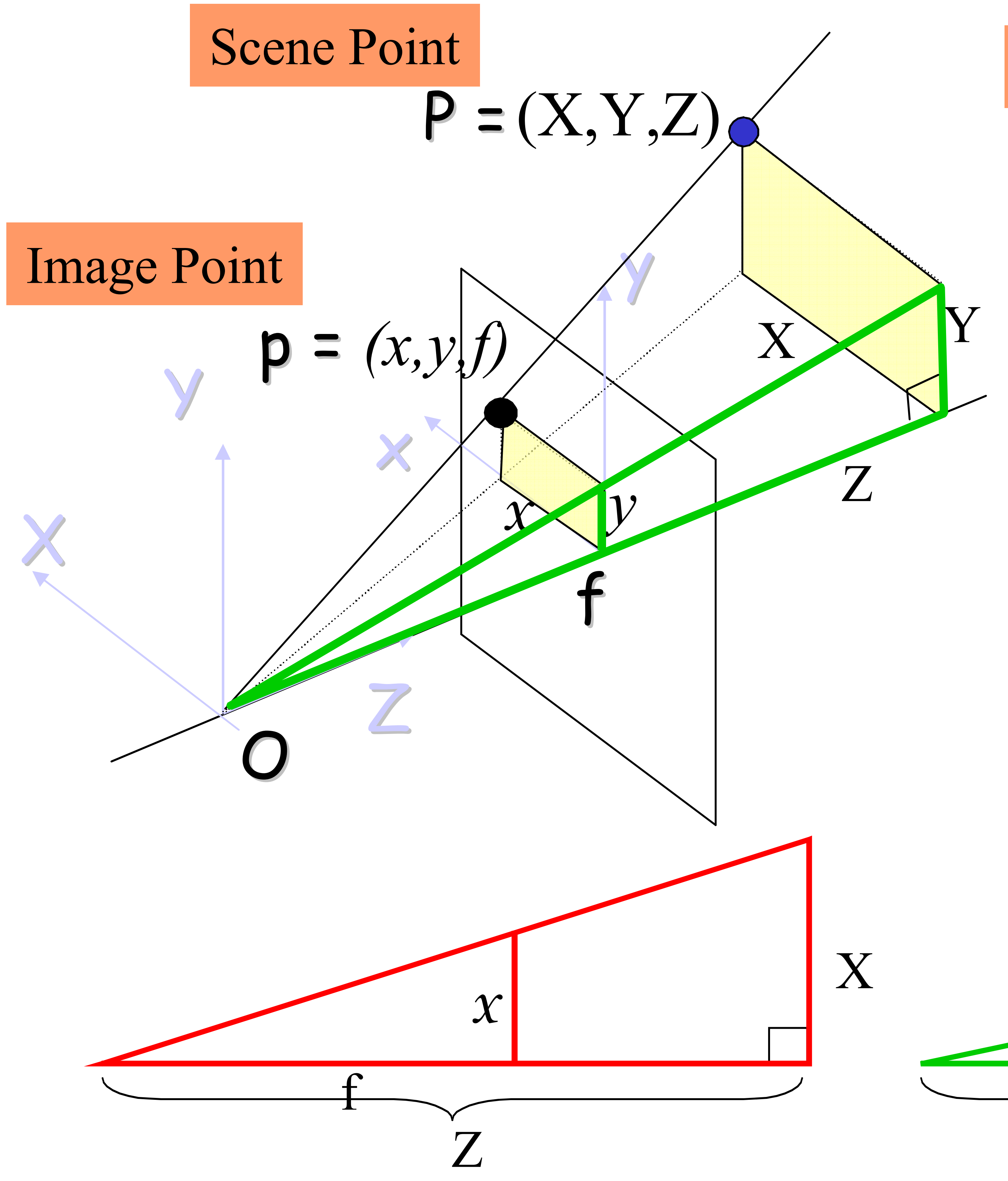
Camera Projections

- **Perspective projection:** transformation of a point in the real world into a point on the camera image frame.
- x, y relations between scene and camera points.
- Z is the depth/distance of the point.

Perspective Projection Eqns

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$



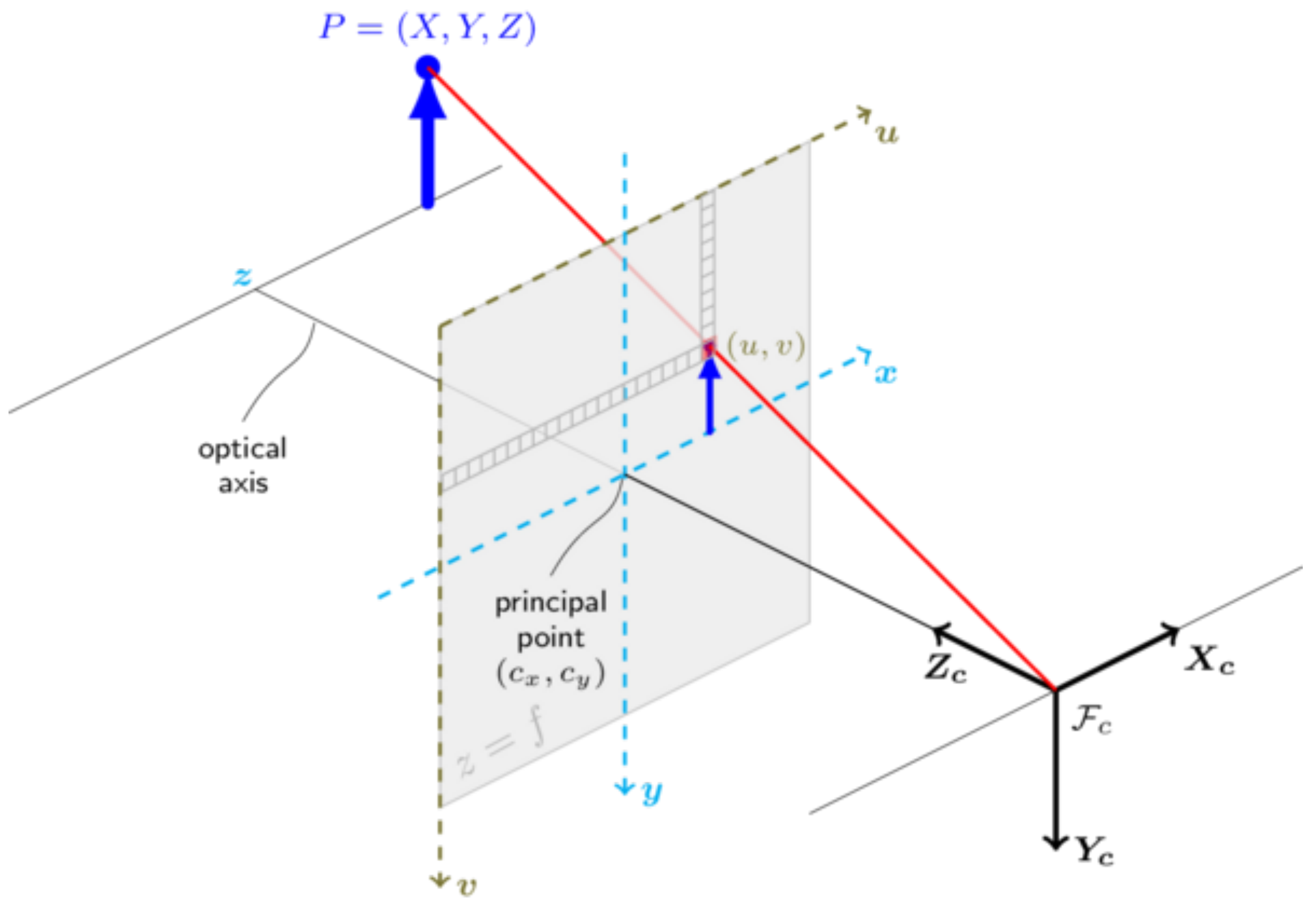
Slide image from: Collins R. Penn State CSE486

Camera Calibration

- How find f, the camera's focal length and other camera parameters?
- Calibration:
  - Intrinsic — estimate internal camera parameters, e.g. focal length
  - Extrinsic — estimate object's 3D pose from 2D point correspondences
- Tools
  - OpenCV calibration API
  - Patterns (checkerboard)

Camera Calibration : Exercise

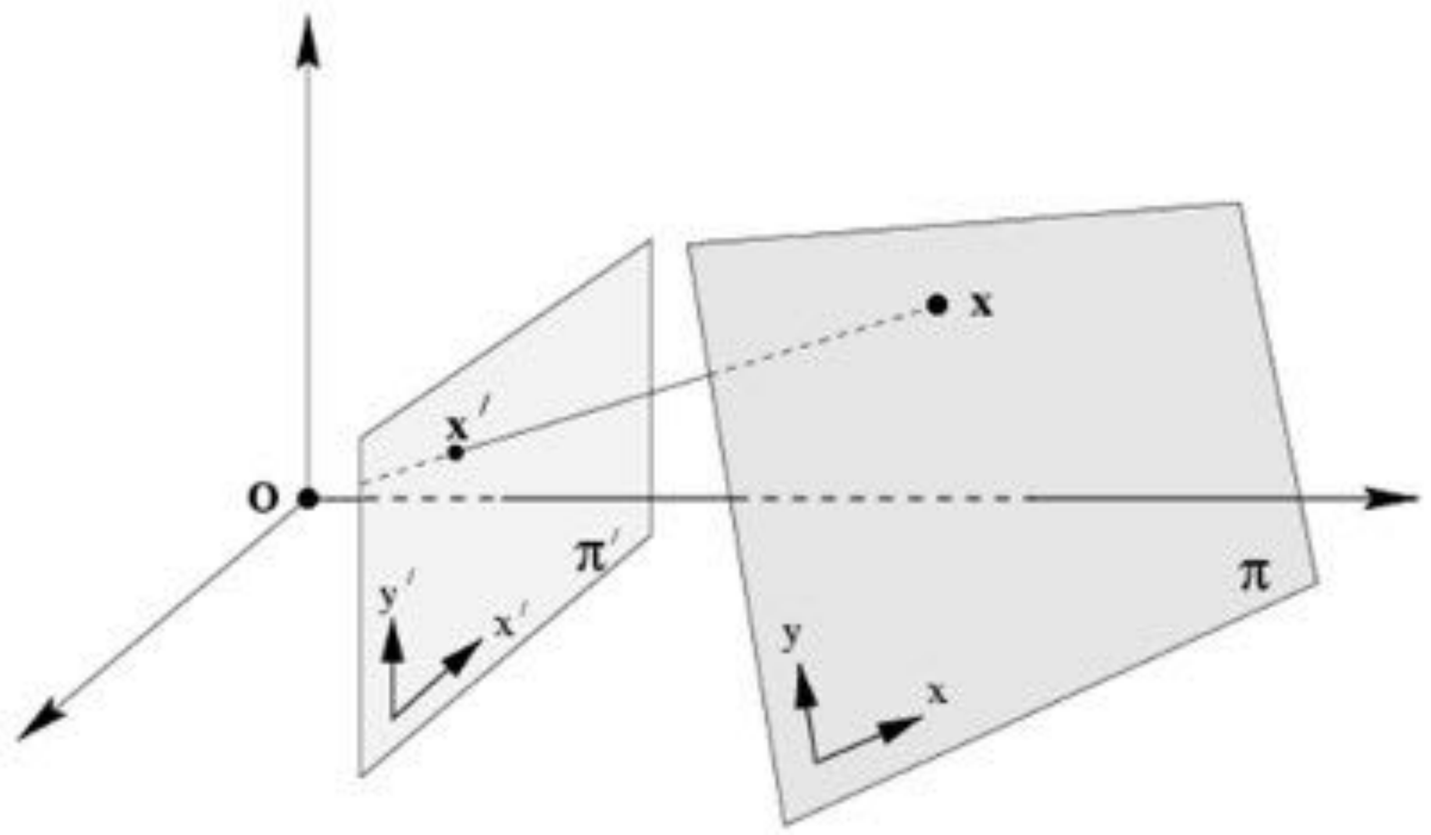
- Calibrate the webcams using OpenCV calibration API.
- We use a pin-hole camera model.
- Hint:
    - Use the provided `stereovision` library.

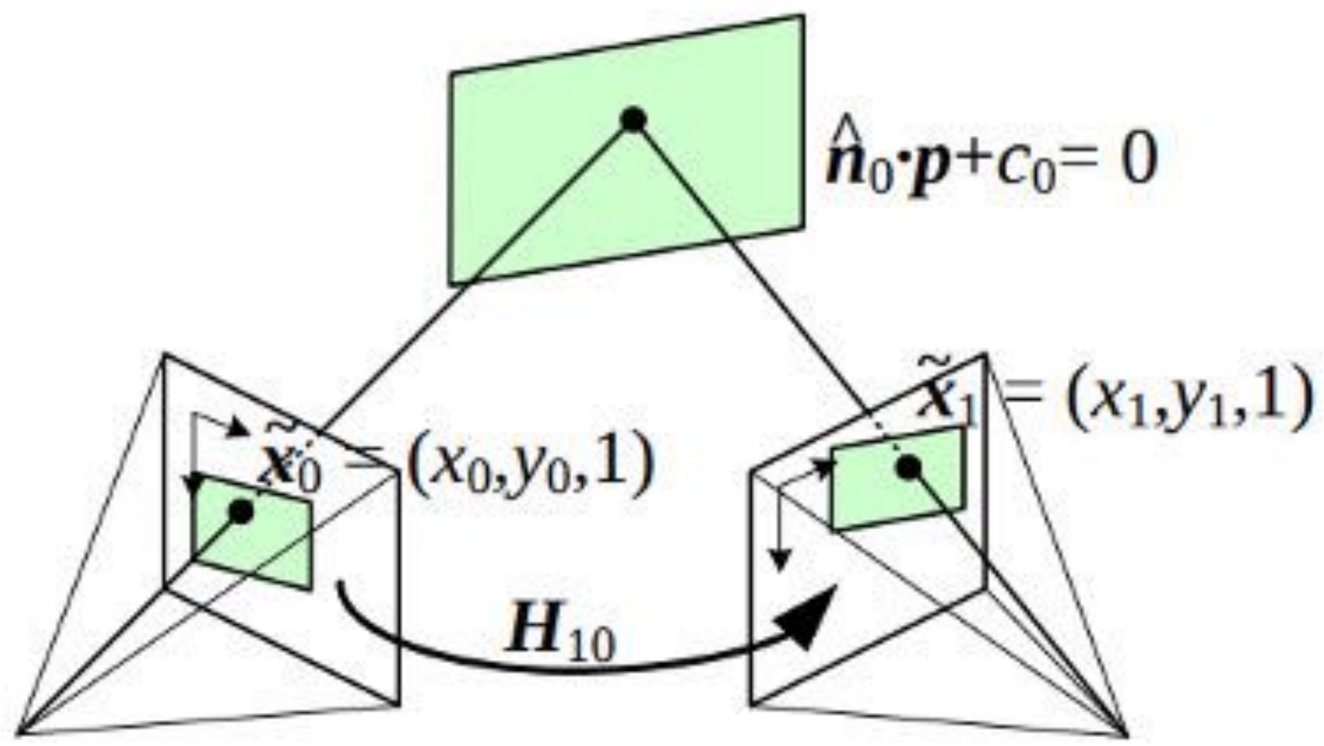


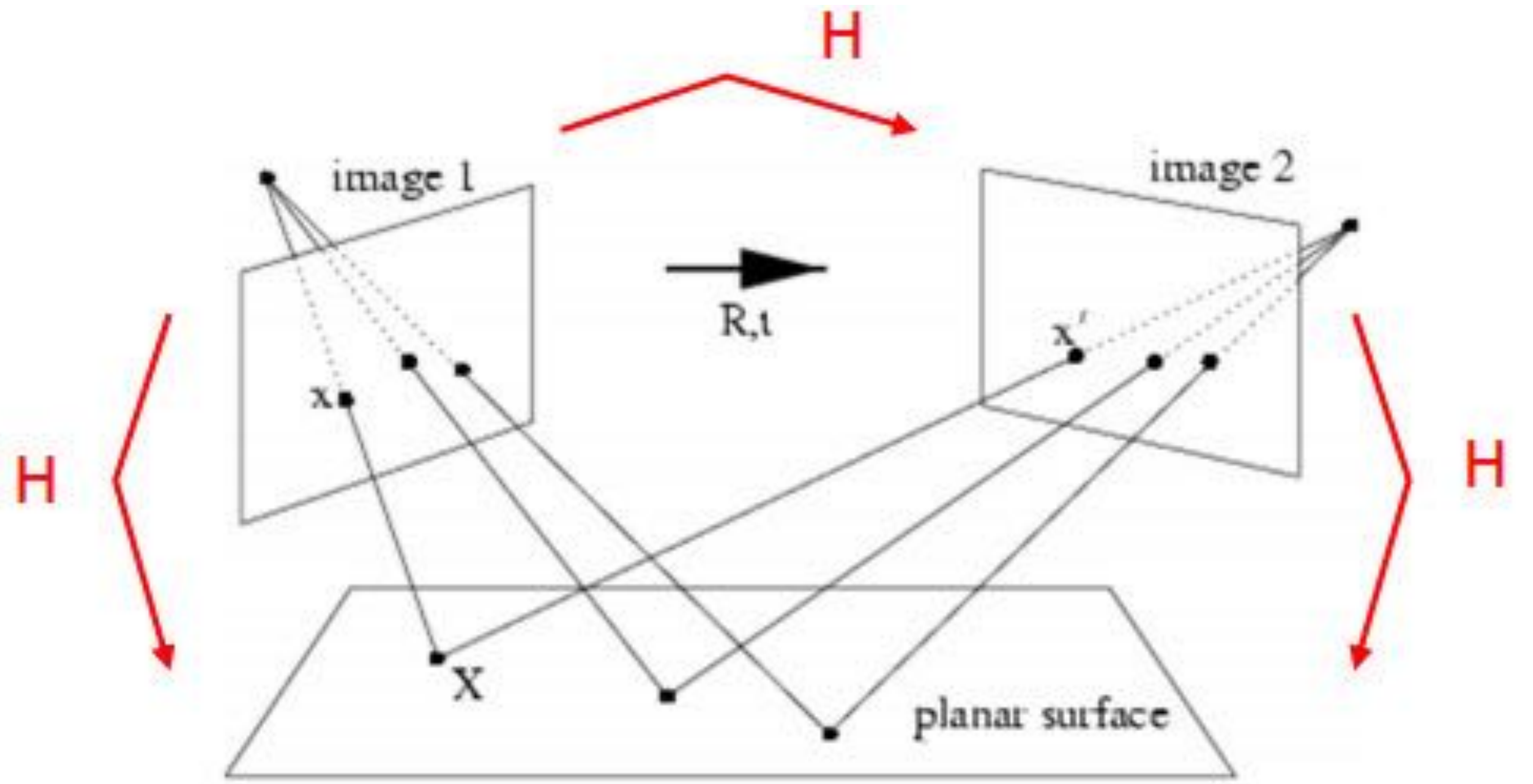
Camera Projections

- **Homography** is a transformation that relates points between two planes (excluding the scale factor)

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$





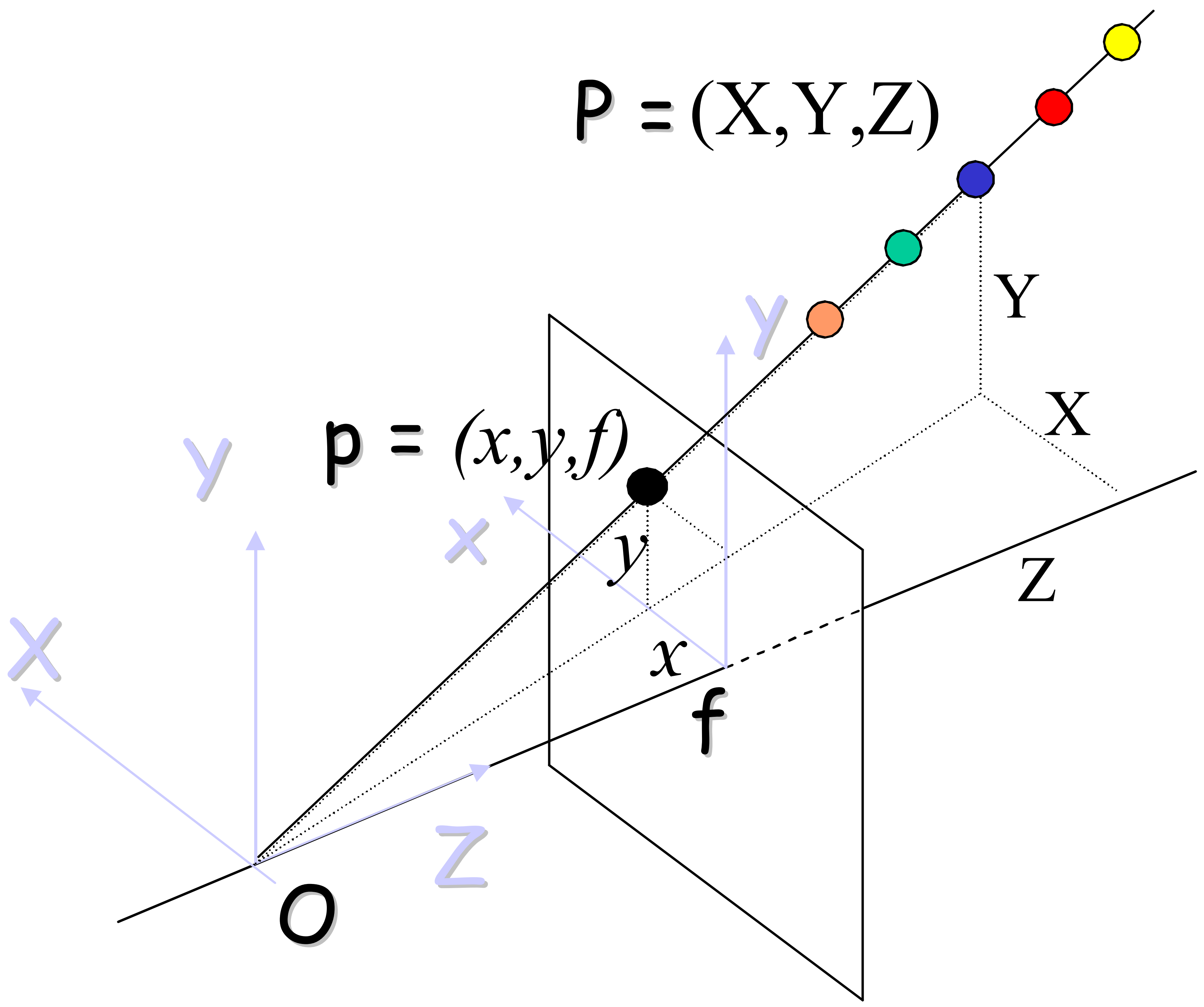


Stereo Vision

Why Stereo Vision?

- Need to disambiguate along the depth axis.

- If we stick to a single camera



$$x = f\frac{X}{Z} = f\frac{kX}{kZ}$$

$$y = f\frac{Y}{Z} = f\frac{kY}{kZ}$$

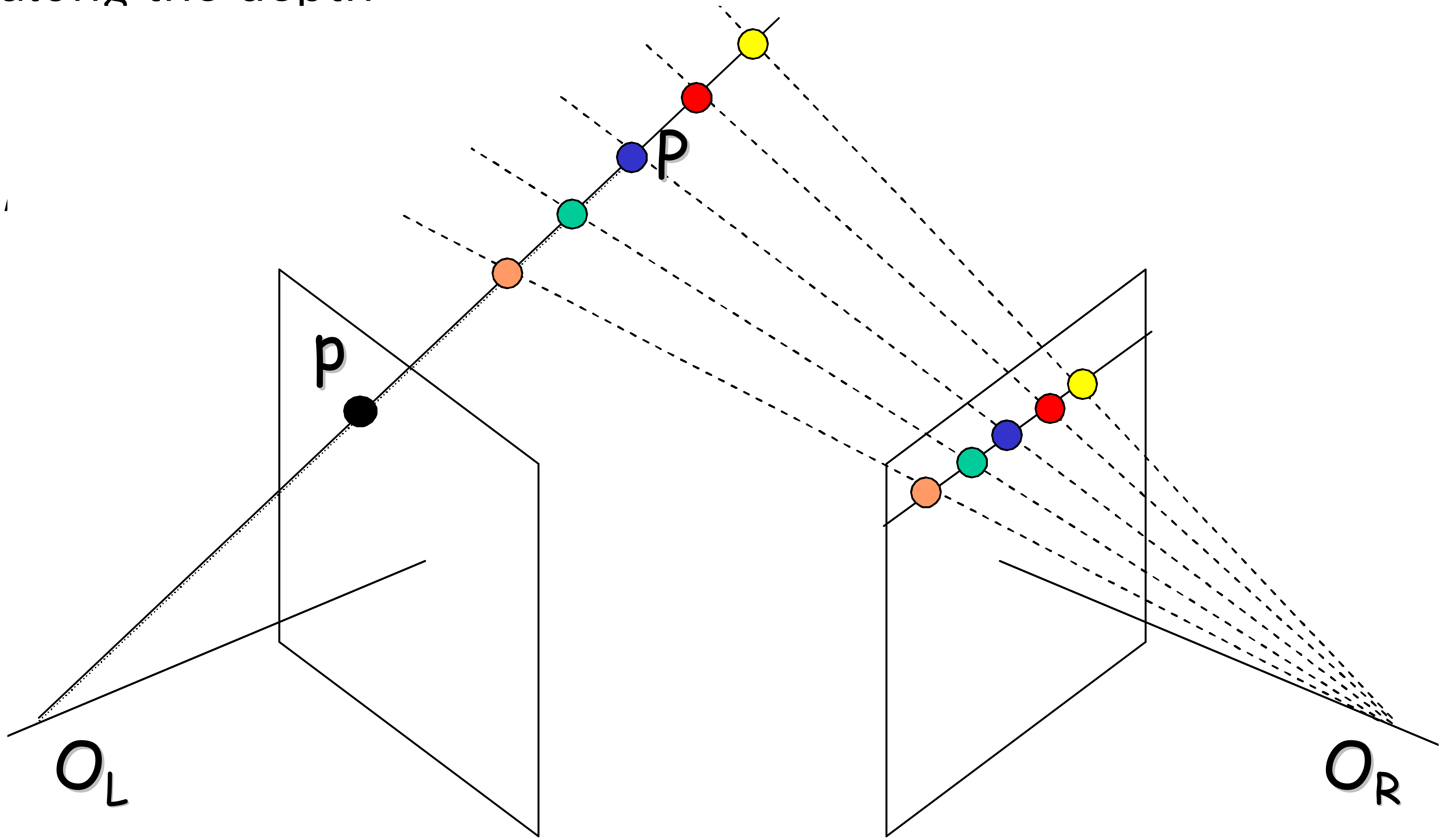
Fundamental Ambiguity:

Any point on the ray OP has image p

Slide image from: Collins R. Penn State CSE486

Why Stereo Vision?

- Need to disambiguate along the depth axis.
- Switch to two cameras, then we can 'see' the depth.
  - Via triangulation



Slide image from: Collins R. Penn State CSE486

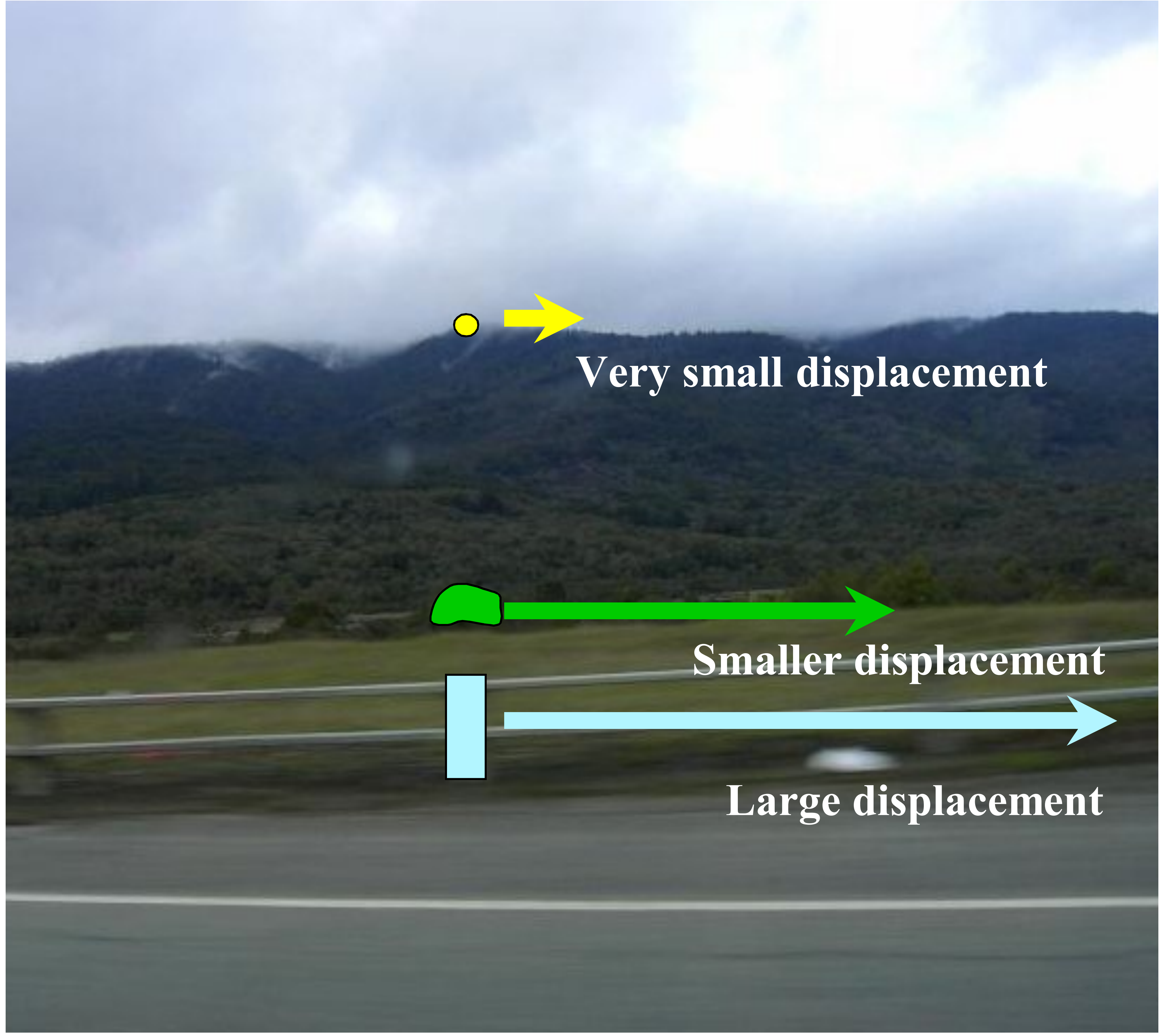
Stereo Vision Intuition

- Parallax

https://vimeo.com/28709243

Stereo Vision Intuition

- Measure parallax to determine distances



Far ⟶ small displacement

Mid ⟶ mid displacement

Close ⟶ large displacement

Slide image from: Collins R. Penn State CSE486

Stereo Vision Estimation

- OpenCV API

Resources

- https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
- https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_face_detection.html
- https://theasciicode.com.ar/ascii-control-characters/escape-ascii-code-27.html

Note

- Blogs
  - Keep daily logs
- Report
  - Due at the end of the week.
- Exercises
  - All count towards final assessment
- Group poster, explaining the project.