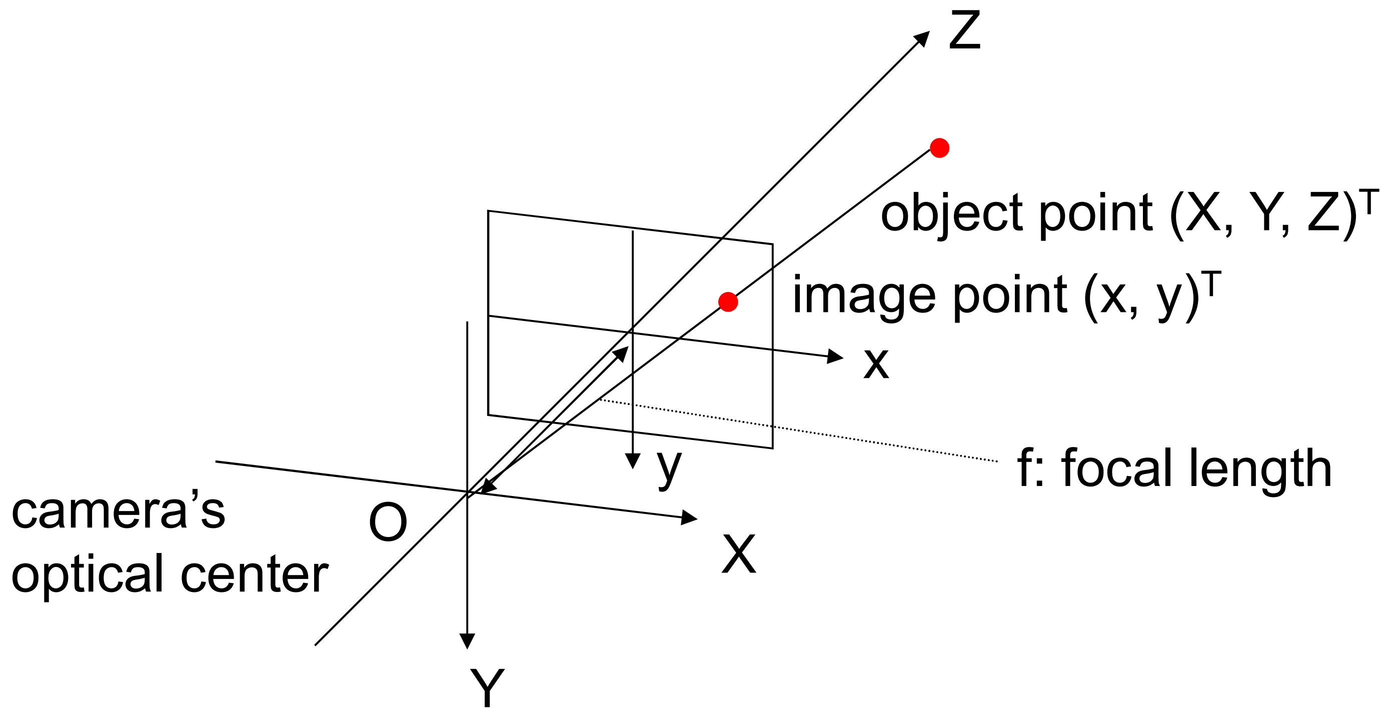
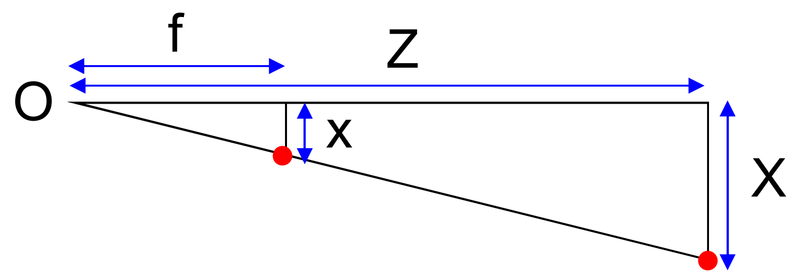
Outline

- Introduction
- Theory
- Software

Pinhole Camera Model

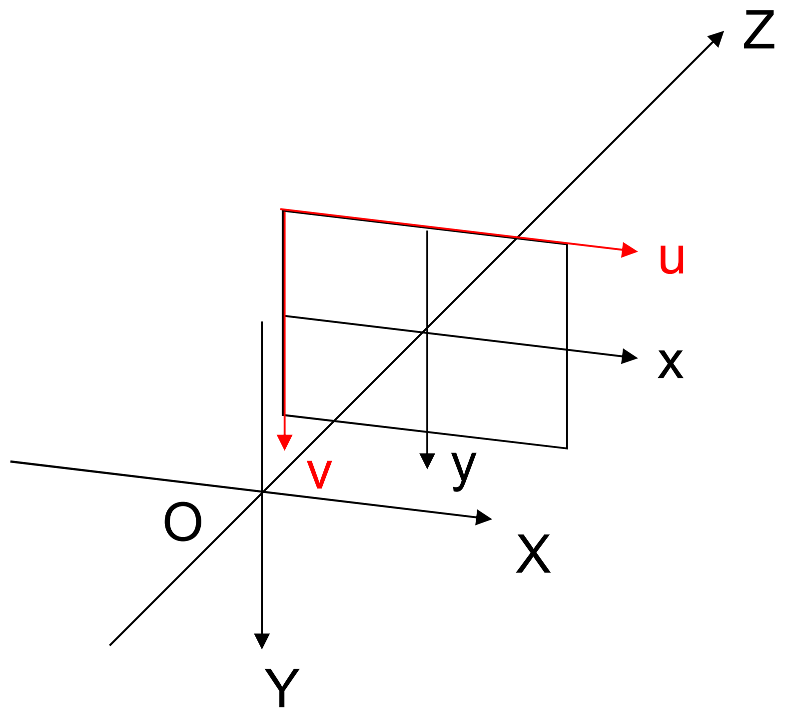


$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}$$



X-Y-Z: camera coordinate frame
x-y: (normalized) image coordinate frame

Pixel Coordinates



$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} k_u x + c_u \\ k_v y + c_v \end{pmatrix}$$

u-v: image coordinate (pixel coordinate) frame

Camera Intrinsic Matrix

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X \\ Y \end{pmatrix} \qquad \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} k_u x + c_u \\ k_v y + c_v \end{pmatrix}$$

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f k_u & 0 & c_x \\ 0 & f k_v & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$= \boxed{A} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

3x3 camera intrinsic matrix

Coordinate Transformation of the Object Point

- Say, the object point coordinates are expressed in a different coordinate frame X'-Y'-Z'

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} + \boldsymbol{t}$$

R: 3x3 rotation matrix
t: 3D translation vector

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} R & | & \boldsymbol{t} \end{pmatrix} \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix}$$

homogeneous coordinate of $(X', Y', Z')^T$

Camera Projection Matrix

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} R & | & \boldsymbol{t} \end{pmatrix}\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix}$$

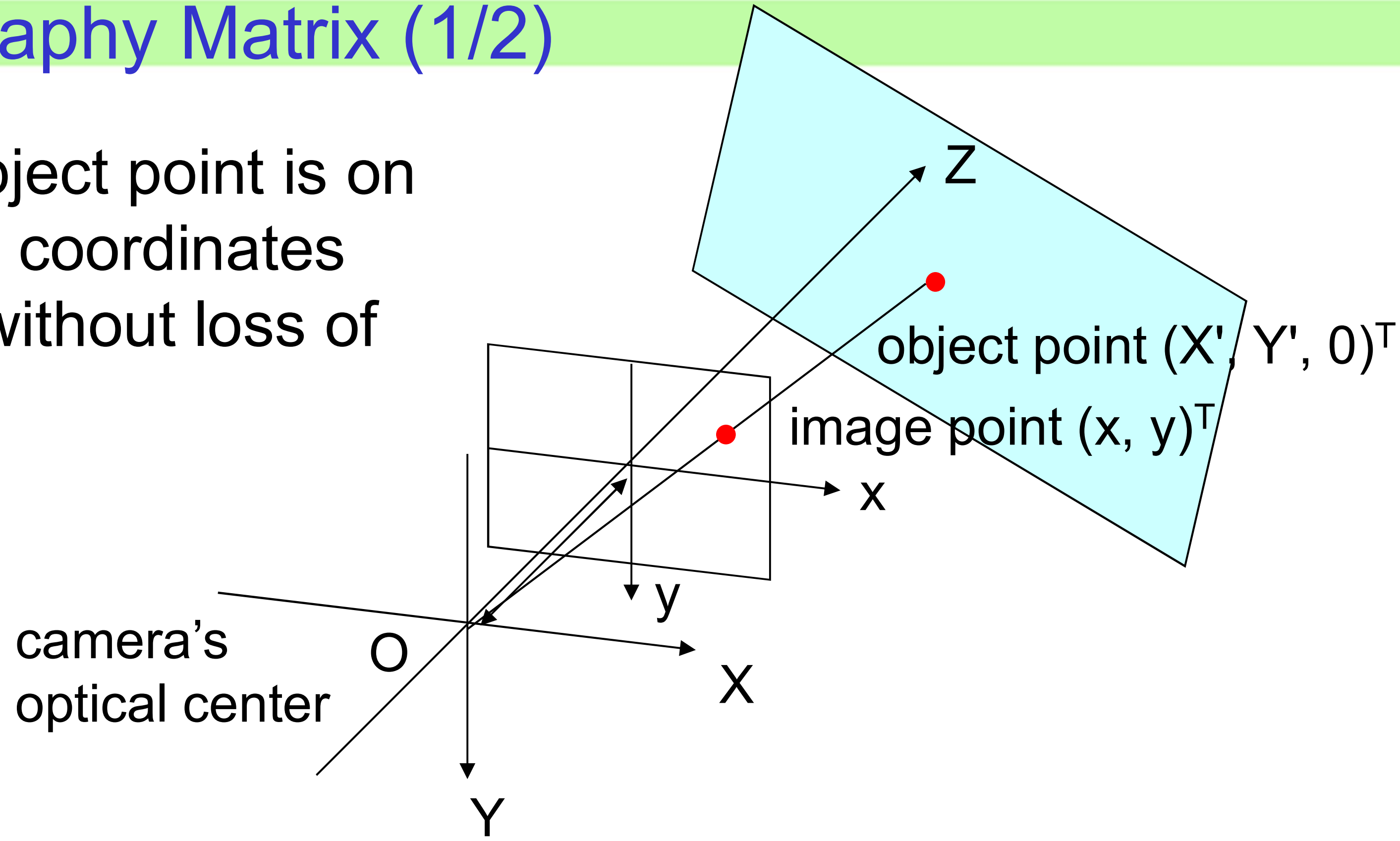
$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A\begin{pmatrix} R & | & \boldsymbol{t} \end{pmatrix}\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix}$$

$$= \boxed{P}\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix}$$

3x4 camera projection matrix

Homography Matrix (1/2)

- Assume the object point is on a plane and its coordinates are (X’, Y’, 0) without loss of generality.



$$Z\begin{pmatrix}u\\v\\1\end{pmatrix}=P\begin{pmatrix}X'\\Y'\\0\\1\end{pmatrix}=\boxed{H}\begin{pmatrix}X'\\Y'\\1\end{pmatrix}$$

3x3 homography matrix

Homography Matrix (2/2)

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix}$$

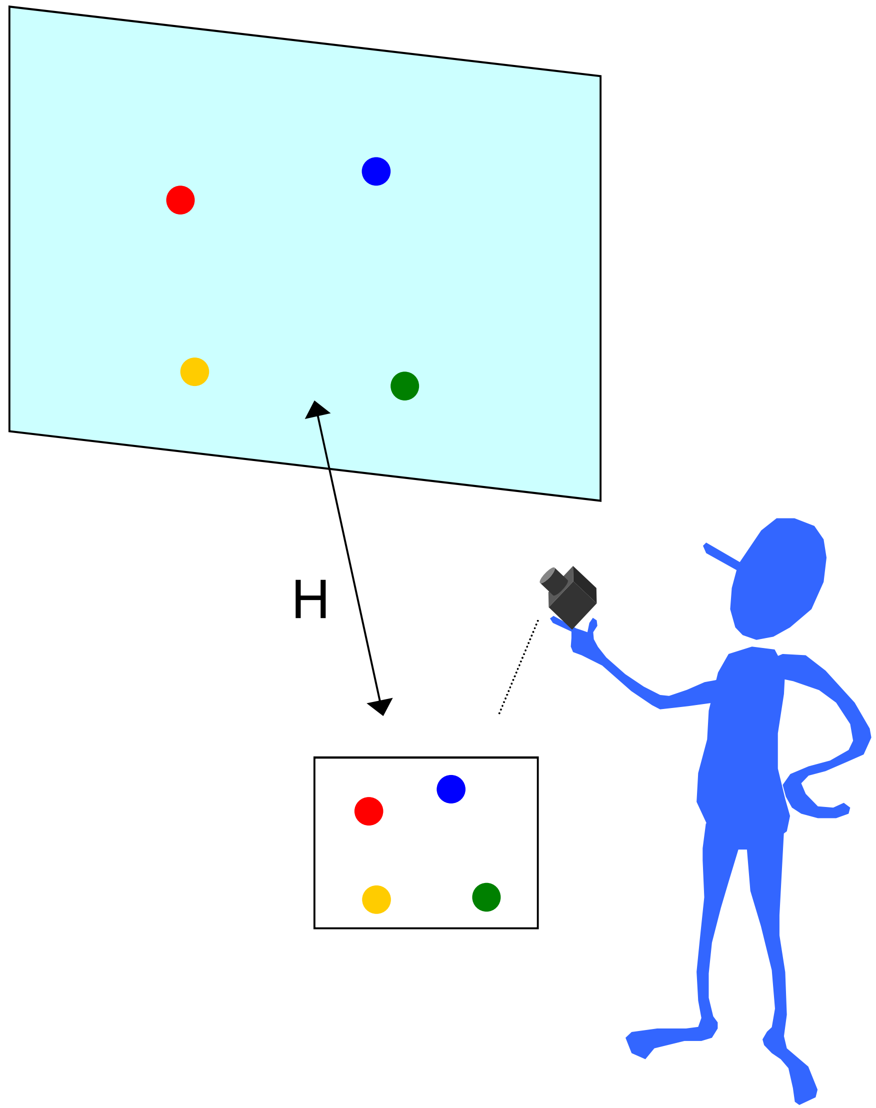
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \simeq H \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix} \qquad \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix} \simeq H^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

- When the homography matrix is given, the mapping from/to (X', Y') and (u, v) are uniquely determined
  - The scale factor Z is determined from the 3rd row eqn.
  - Then the other rows eqns are solved
- When n (>= 4) known points on a plane are observed by a camera, H can be computed

Summary so far

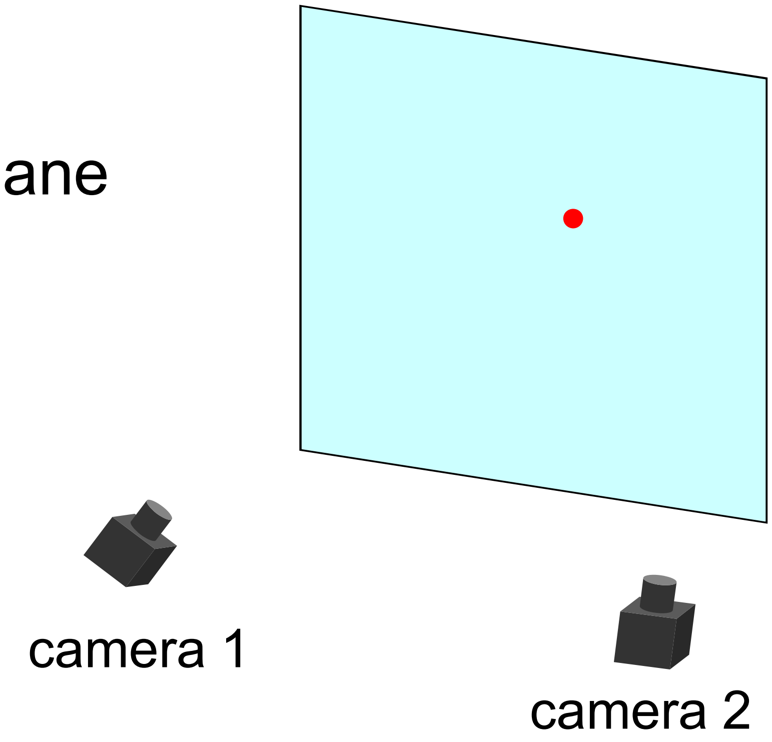
- If you have four or more corresponding points between the screen and the camera image, homography matrix between them can be computed

- Once homography matrix is obtained, any point coordinates can be transformed from screen to camera, and vice versa



Homography Between Two Views (1/2)

- Think of two cameras observing a point on a plane



$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \simeq H_1 \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix} \qquad \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \simeq H_2 \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix}$$

Homography Between Two Views (1/2)

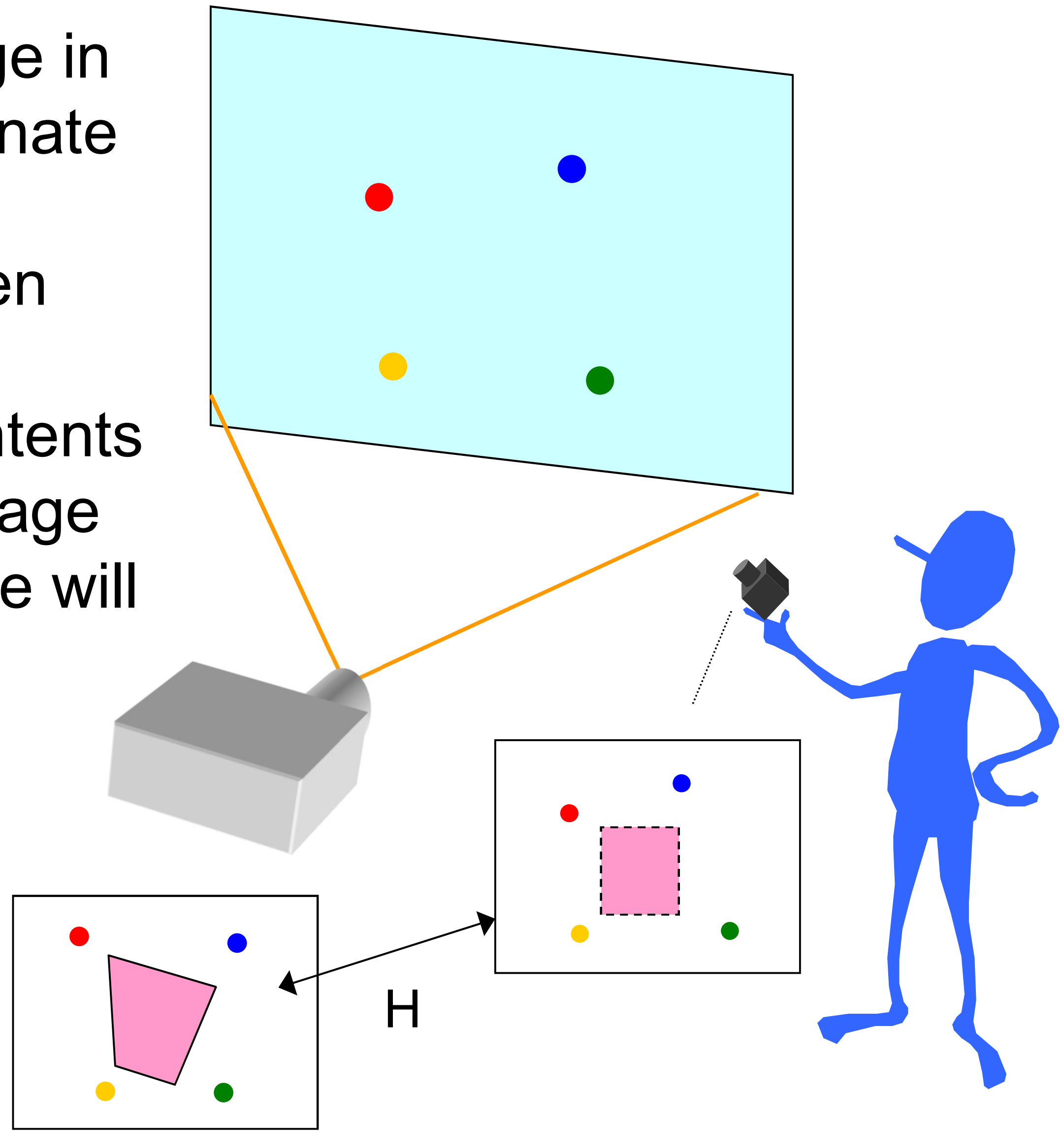
$$\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \simeq \boxed{H_2 H_1^{-1}} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}$$

3x3 homography matrix that gives mapping between the pixel coordinates of camera 1 / camera 2

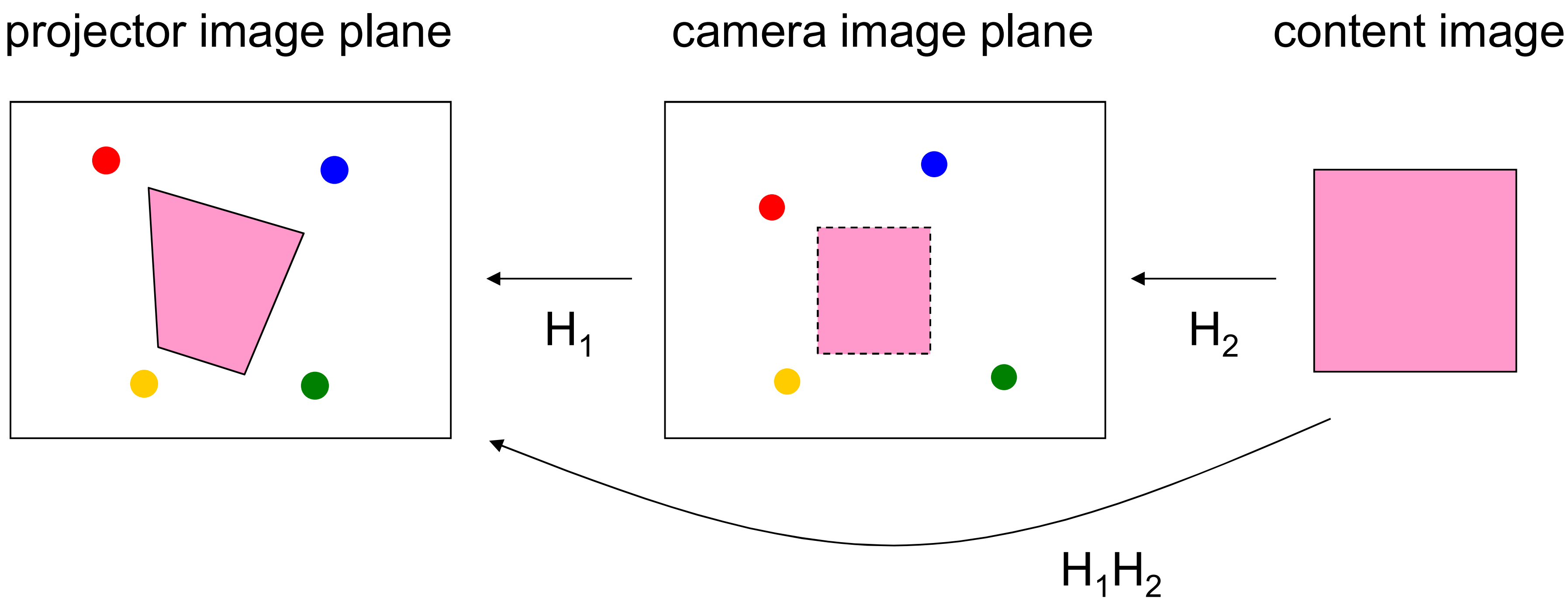
- When n (>= 4) corresponding points on a plane are observed by the two cameras, $H = H_2H_1^{-1}$ can be computed
- Once H is obtained, the mapping from/to camera 1 and camera 2 pixel coordinates is straightforward
- One of the camera can be a projector

So, what to do is...

- (Prepare a contents image in the camera image coordinate frame)
- Find homography between projector and camera
- Transform (warp) the contents image to the projector image
- Then, the projected image will look as if the camera is a handheld projector

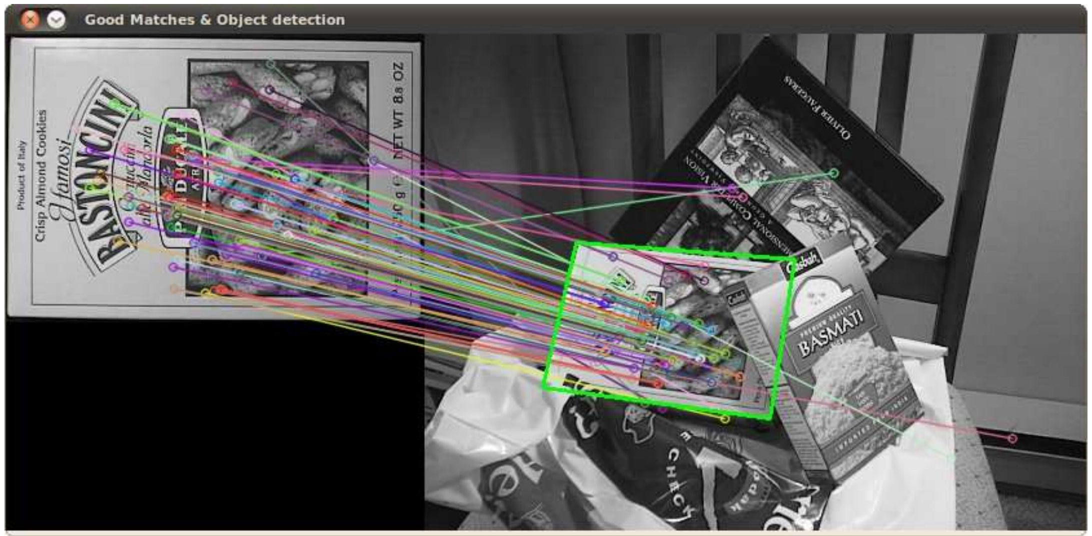


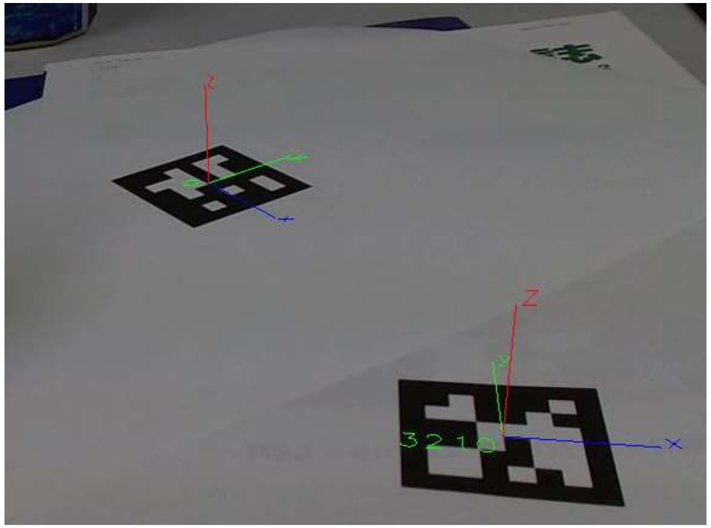
Transitive Application of Homography Matrices



Corresponding Points Problem

- In principle, because we know what kind of image we display from the projector, making correspondence between the points in the projector image and those in the camera image
- But how to do it practically?
  - Local image feature descriptors (Image keypoints)
  - Fiducial markers





Outline

- Introduction
- Theory
- Software

Python Install – OpenCV 3 pre-built for Windows

Install python 2.x or 3.x

- https://www.python.org/downloads/

Windows (pre-built binaries)

- Install pip: http://stackoverflow.com/questions/4750806/how-do-i-install-pip-on-windows

- Pre-built binaries (OpenCV 2.4 & 3.1, python 2.7 & 3.4-3.5): http://www.lfd.uci.edu/~gohlke/pythonlibs/#opencv

- Run in cmd or powershell:
```
pip install [filename_of_pre-built_binares]
```

- Open python IDLE, write these lines to check that it works:
```
import cv2
print(cv2.__version__) # in python 3.x
# print cv2.__version__ in python 2.x
```

Python Install – OpenCV 3 for OSX and Ubuntu

OSX (for Python 3.4 or 2.7)

- http://www.pyimagesearch.com/2015/06/29/install-opencv-3-0-and-python-3-4-on-osx/
- http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/

Ubuntu (for Python 3.4 or 2.7)

- http://www.pyimagesearch.com/2015/07/20/install-opencv-3-0-and-python-3-4-on-ubuntu/
- http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/

Sample Code for python with OpenCV 3

Inside folder python_sample

- background_demo.py – shows an image
- camcapture_demo.py – shows captured webcam frames

OpenCV information

- Official Site
  - http://opencv.org/

- Python-OpenCV Tutorial
  - http://docs.opencv.org/3.2.0/d6/d00/tutorial_py_root.html

  - Geometric Transformations of Images
    - http://docs.opencv.org/3.2.0/da/d6e/tutorial_py_geometric_transformations.html
  - Feature Matching + Homography to find Objects
    - http://docs.opencv.org/3.2.0/d1/de0/tutorial_py_feature_homography.html