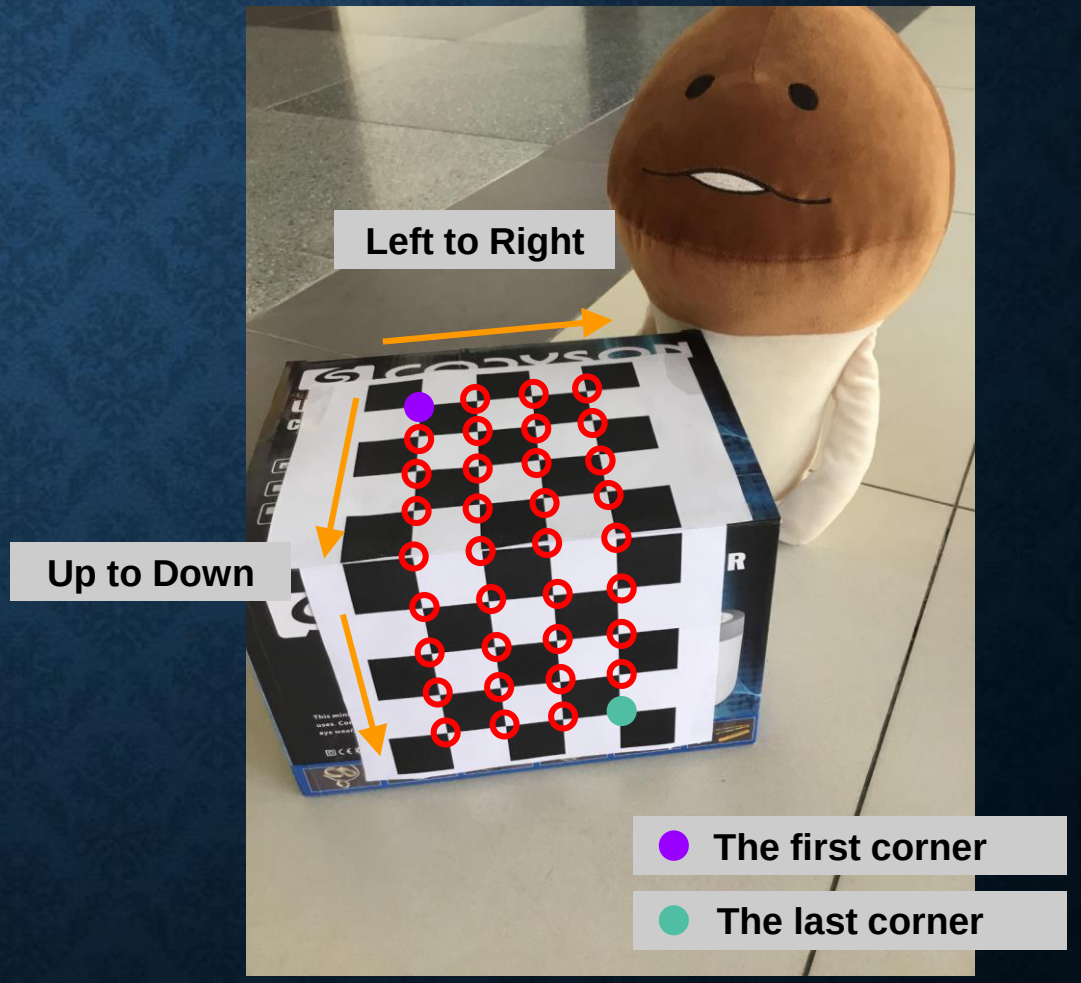
HW2 SUPPLEMENT

Camera Calibration

Label corners of the chessboard

- Use “clicker.py” released by TA or implement by yourself
- 3D points are given in “Point3D.txt”
- Label 36 corners in the order same with the illustration here
  - Total: 36 corners * 2 images
- 1-A compute projection matrix

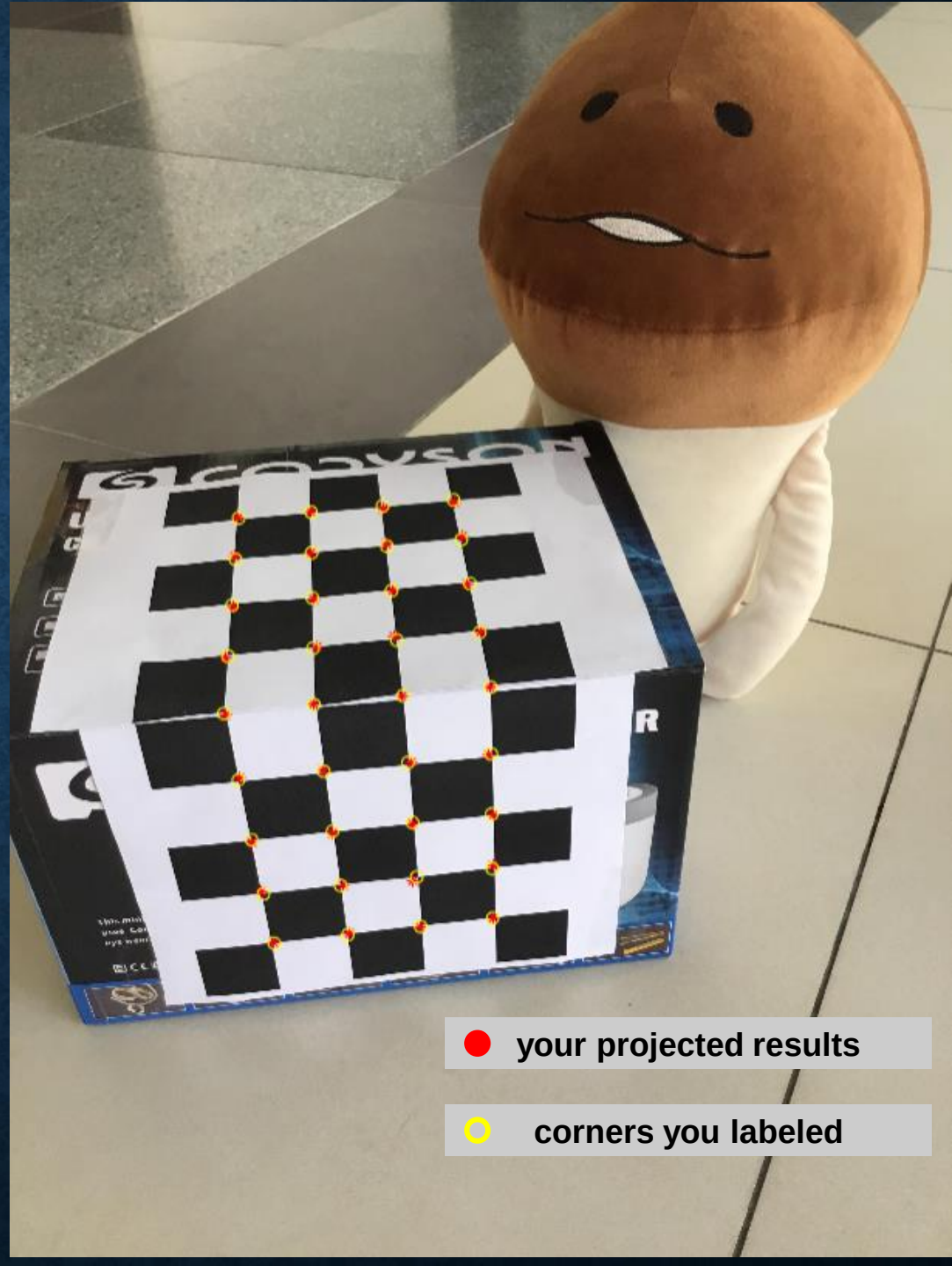


Part 1-B hint

- $P = K [R \mid t]$,
  - K is an upper triangular matrix, and R is an orthogonal matrix
- QR decomposition
  - Q is an orthogonal matrix, and R is an upper triangular matrix

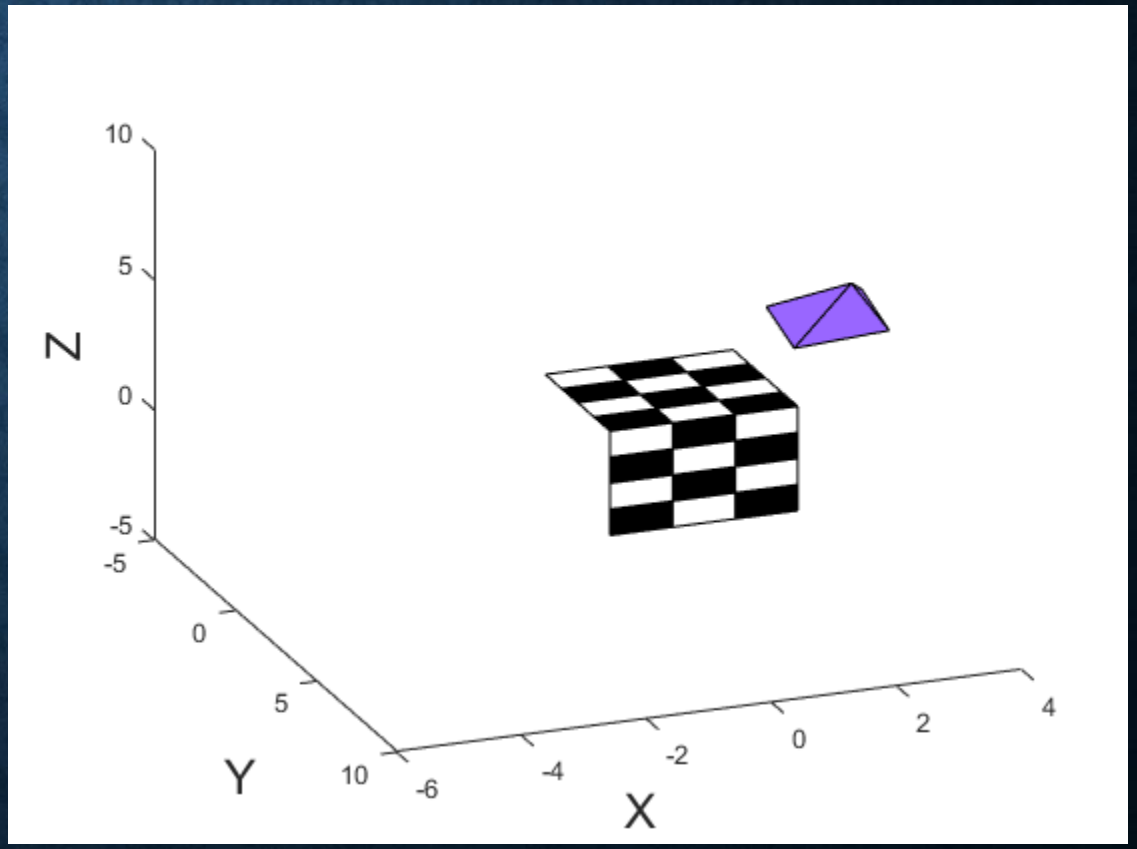
Part 1-C

- Compute RMS error & re-project 2D points on both chessboard images



1-D Visualize camera positions

- You need to visualize the camera positions of both images. We just give the illustration for single camera position here.



Homography transformation

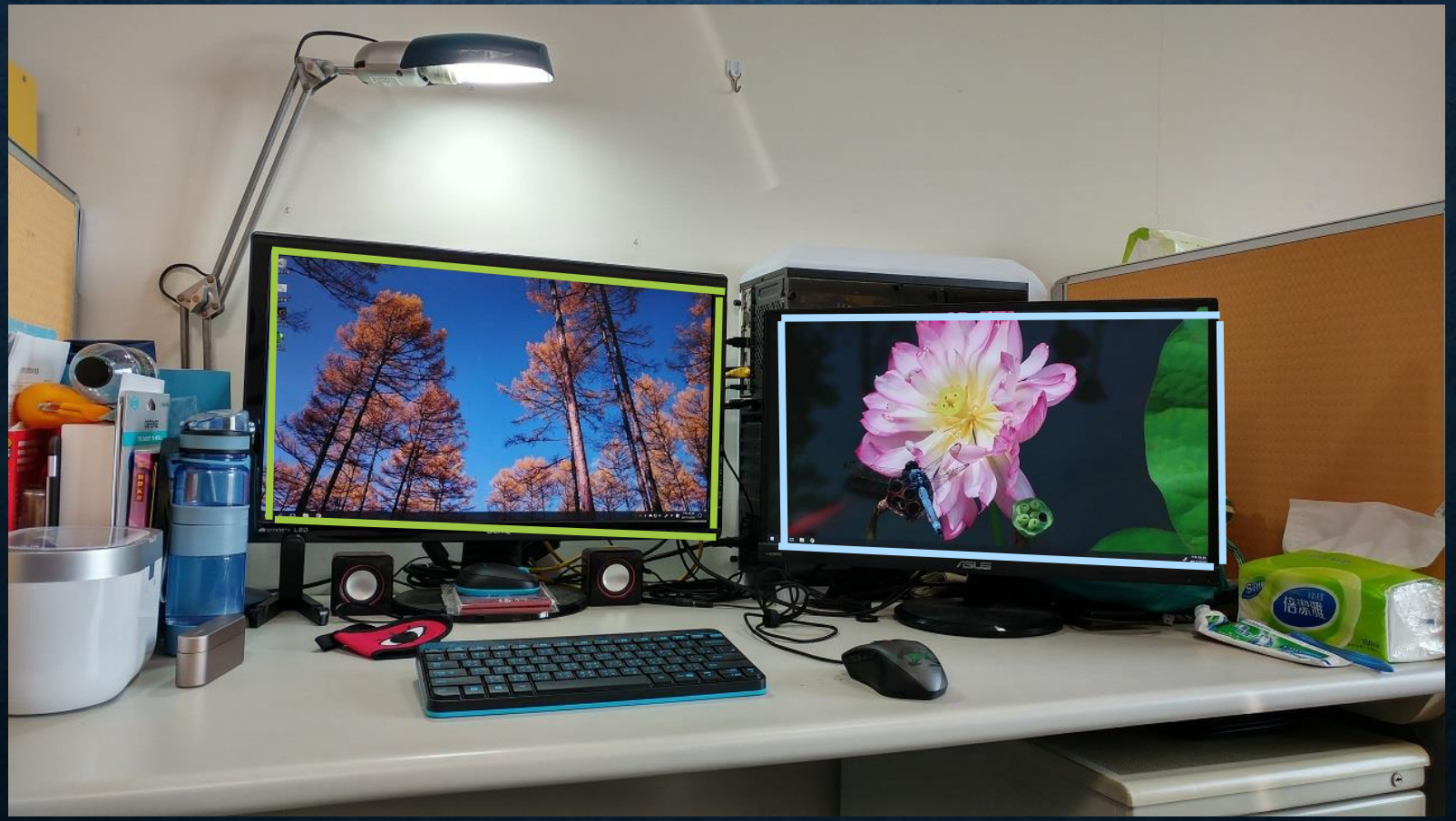
&

Image Warping

What me want to do:
Plane Switch

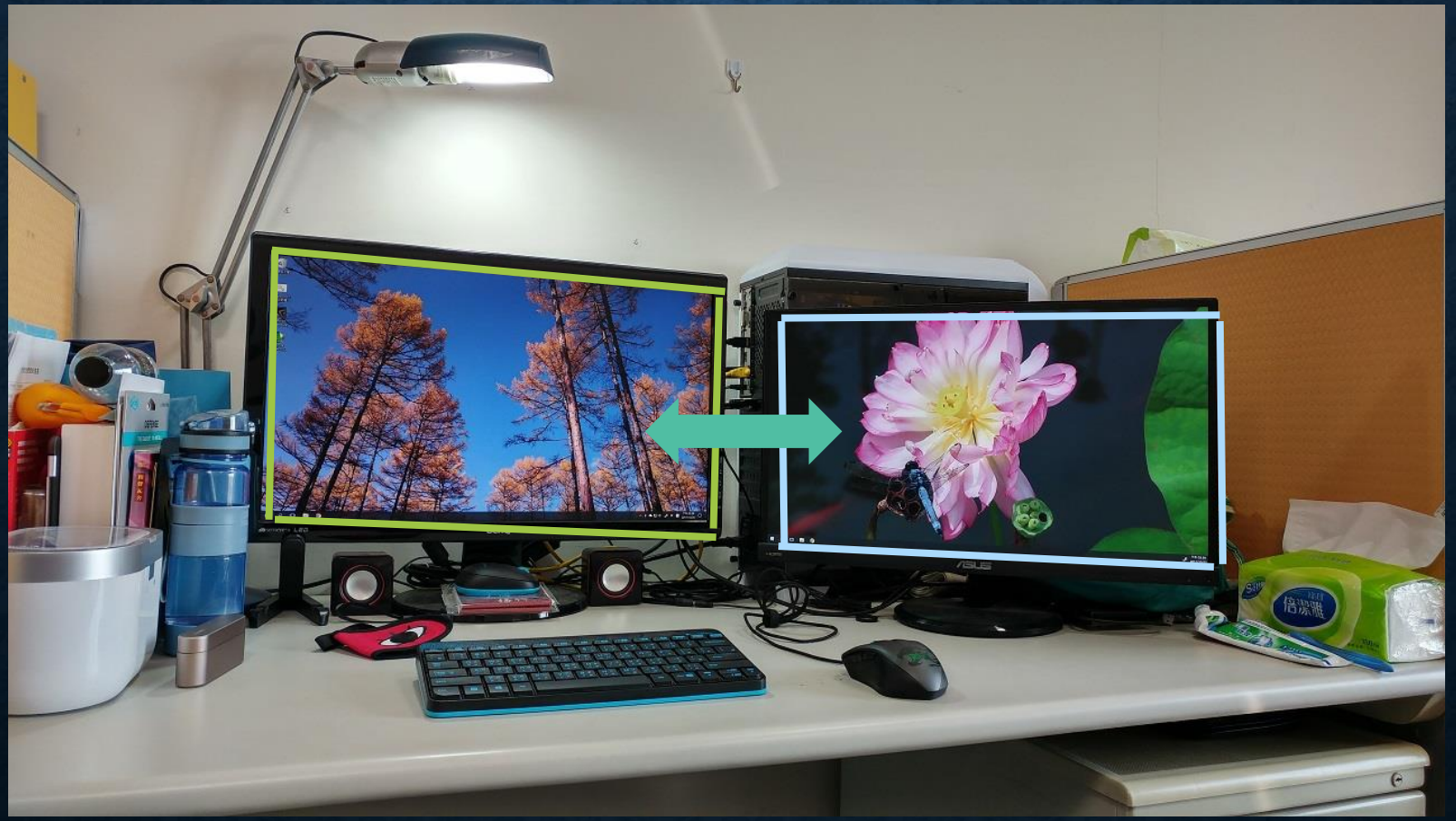
Instruction

1. Find the homography transformation between two planes



Instruction

2. Switch the planes with transformation through image warping



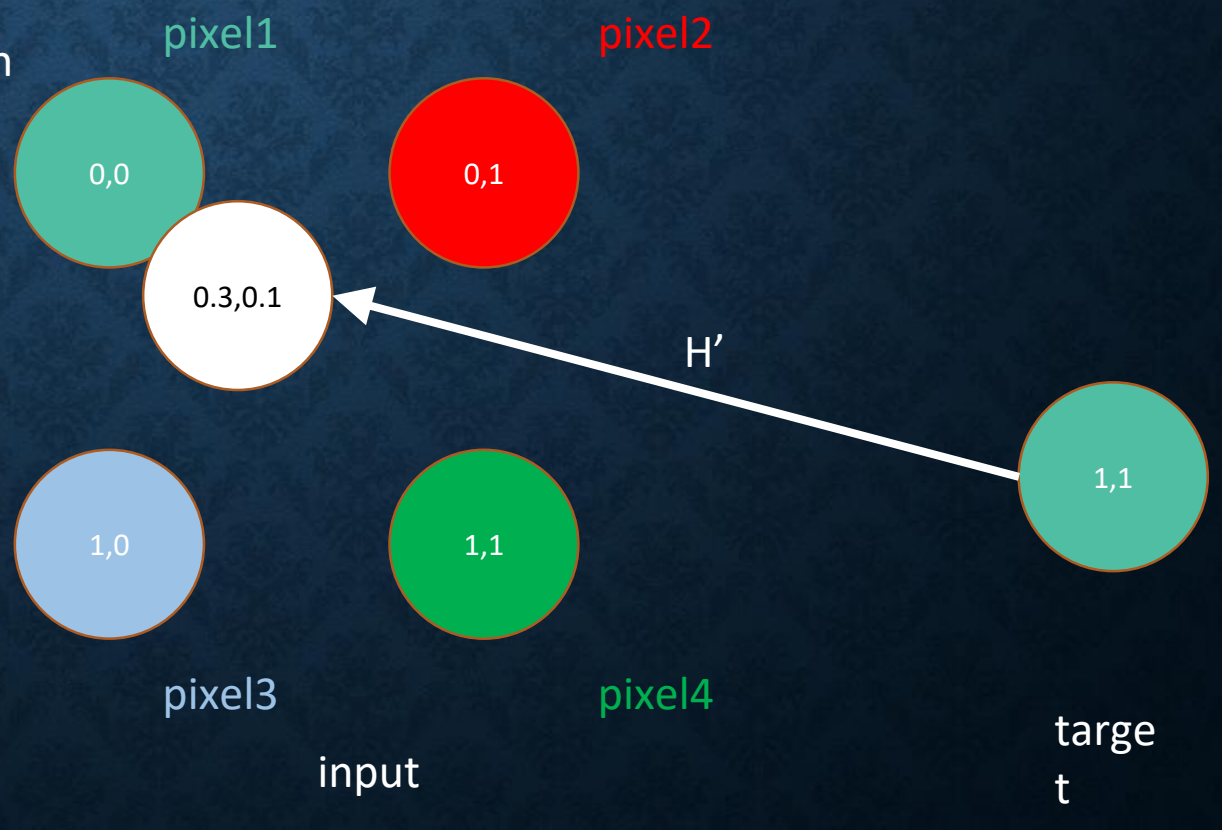
Backward warping

Nearest neighbor backward warping

Search what pixels produce you.

Ex. target pixel comes from input image location

(0.3,0.1), so its color is the nearest one (0,0)



Forward warping

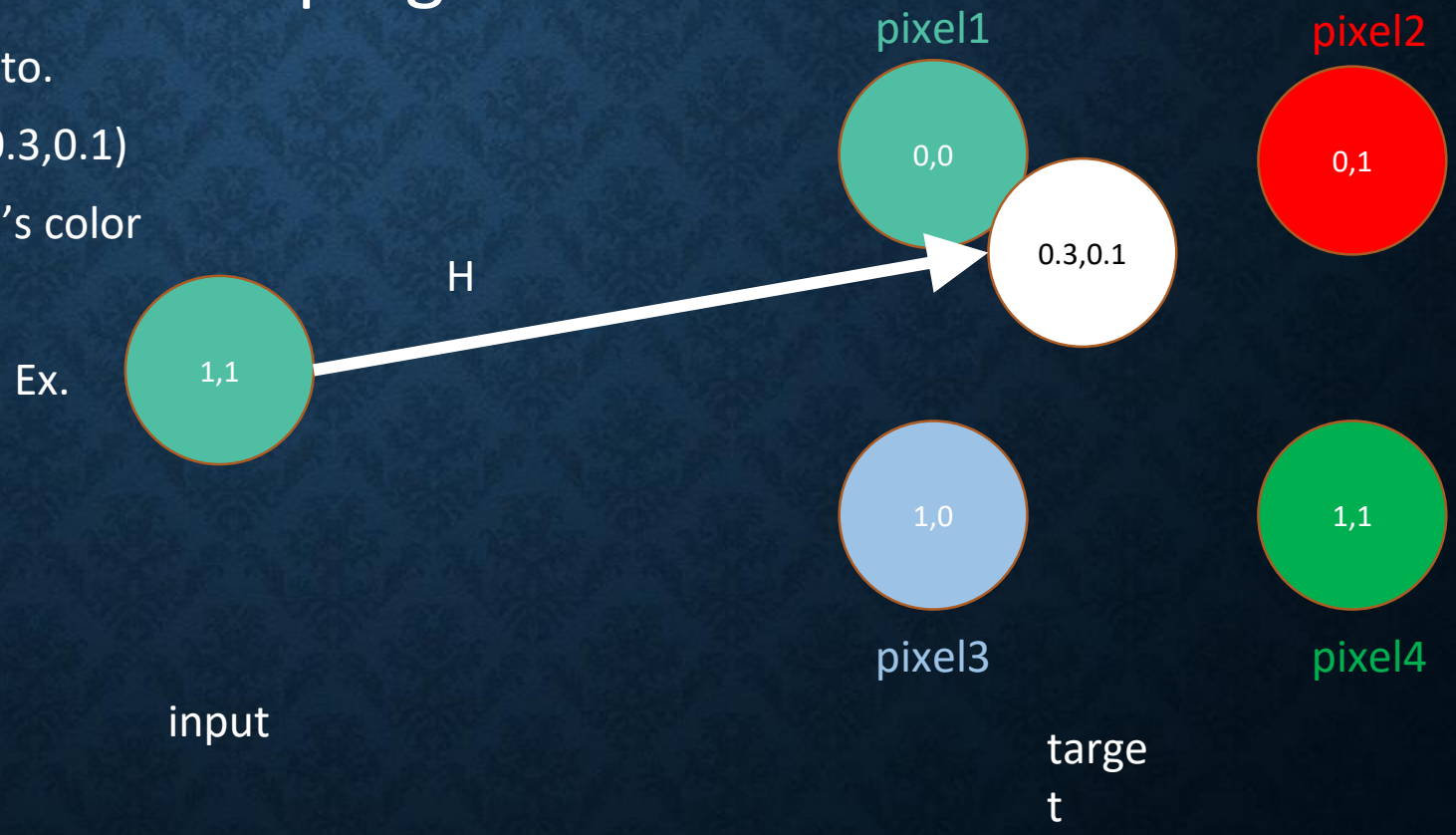
Nearest neighbor forward warping

You need to find pixels you offer color to.

Input pixel transform to the location (0.3,0.1)

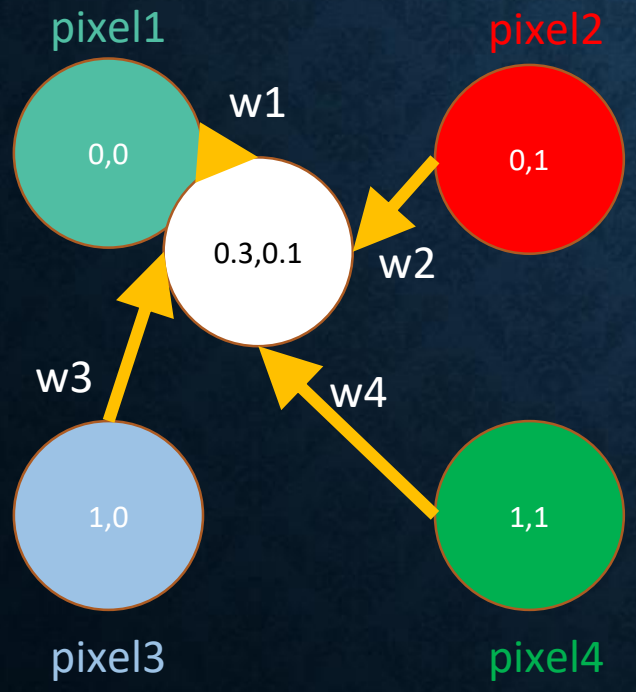
and pixel1 is the nearest one, so pixel1's color

fill with input pixel color



Bilinear Interpolation

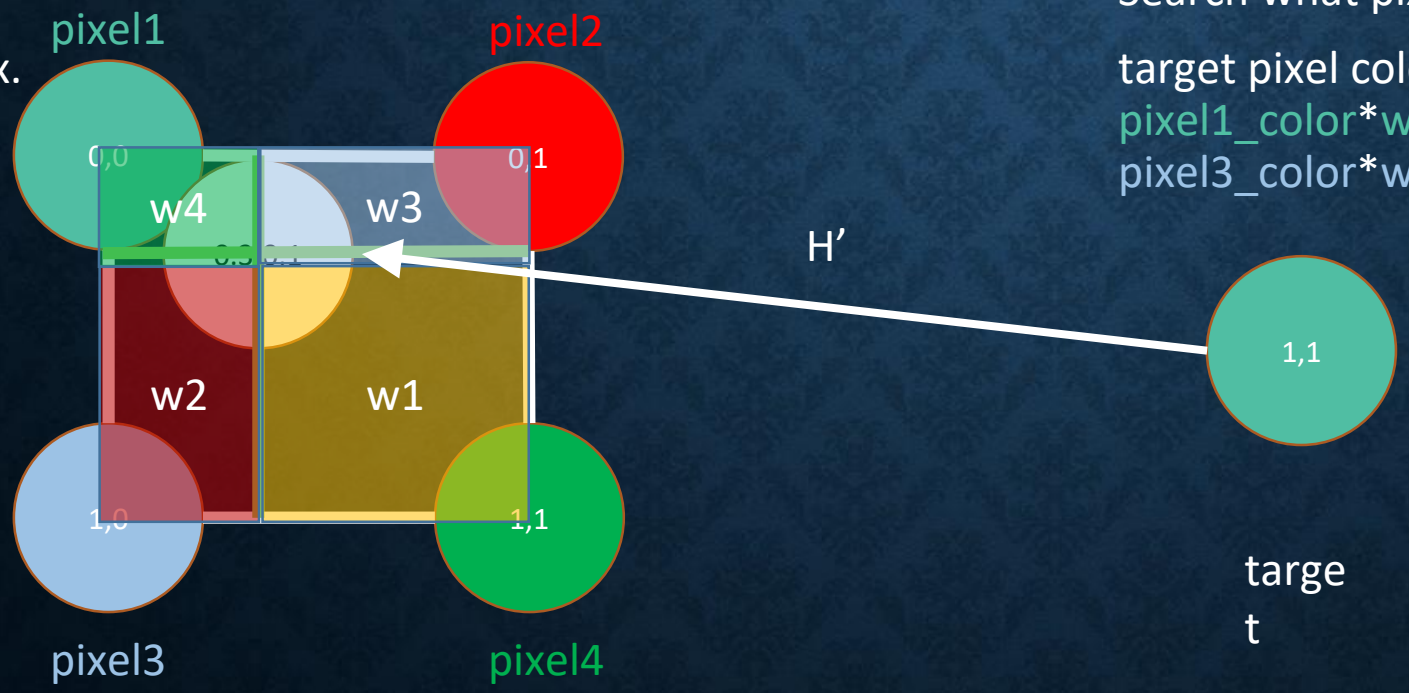
- You need to consider the 4 near pixels to generate (interpolation) your color.
- Weight w is computed by its range.
- Color = pixel1_color*w1+pixel2_color*w2+pixel3_color*w3+pixel4_color*w4



Backward warping

Bilinear backward warping

Ex.



Search what pixels produce you.

target pixel color is
pixel1_color*w1+pixel2_color*w2+…
pixel3_color*w3+pixel4_color*w4