# HW2 Supplement

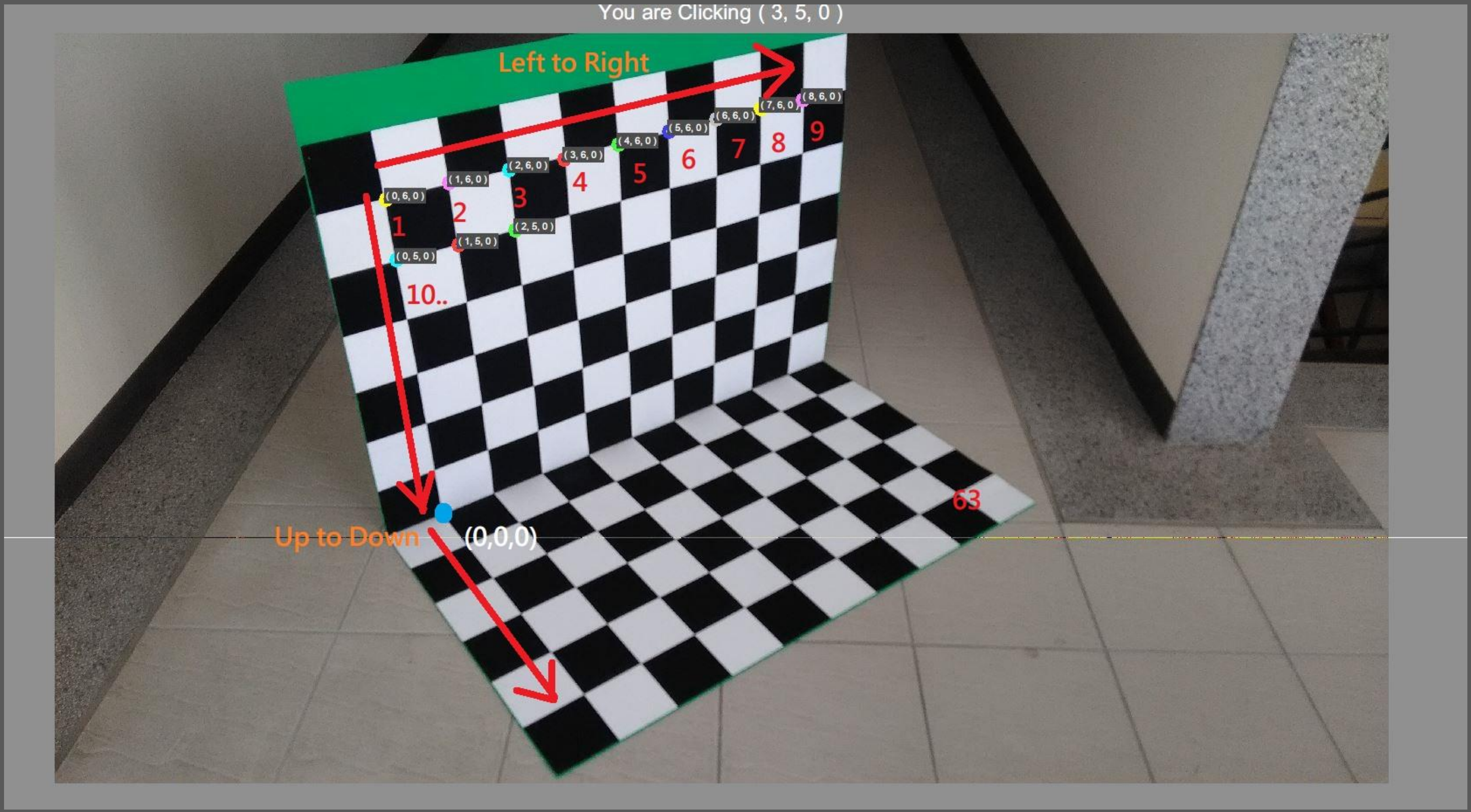
# Detect corners of chessboard

Use Clicker.m provided by TA

1.maximize your window

2.Click corners in order

Total : 108 points * 2 images



You are Clicking ( 3, 5, 0 )

# Part1-B

P = K [R | t],

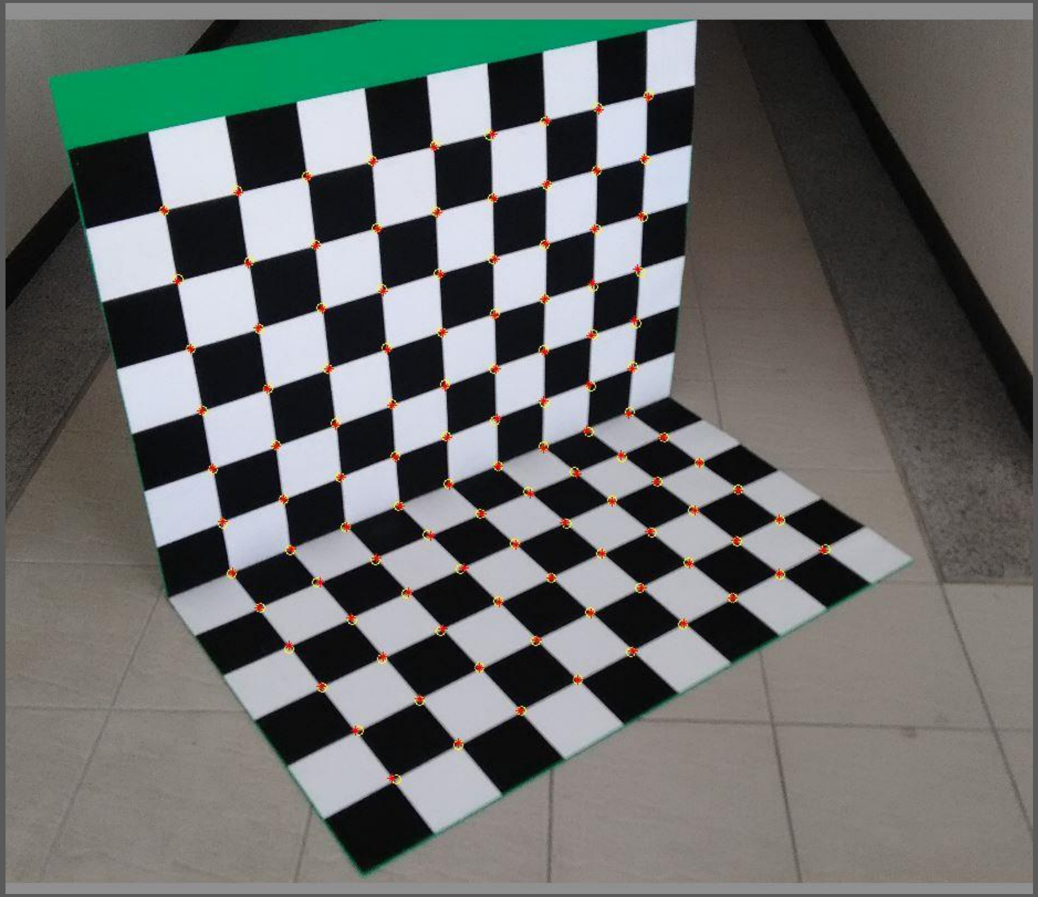
-> K is upper triangular matrix and R orthogonal matrix

QR decomposition

-> Q orthogonal matrix and R upper triangular matrix

# Part1-C

## Compute RMS error & Re-project 2D points on the “chessboard.jpg”

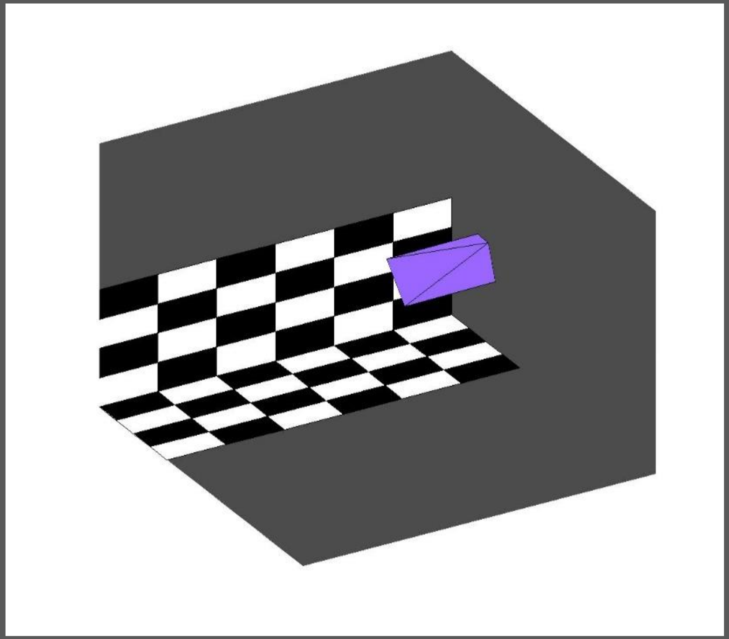


* your predict result

O is your click data

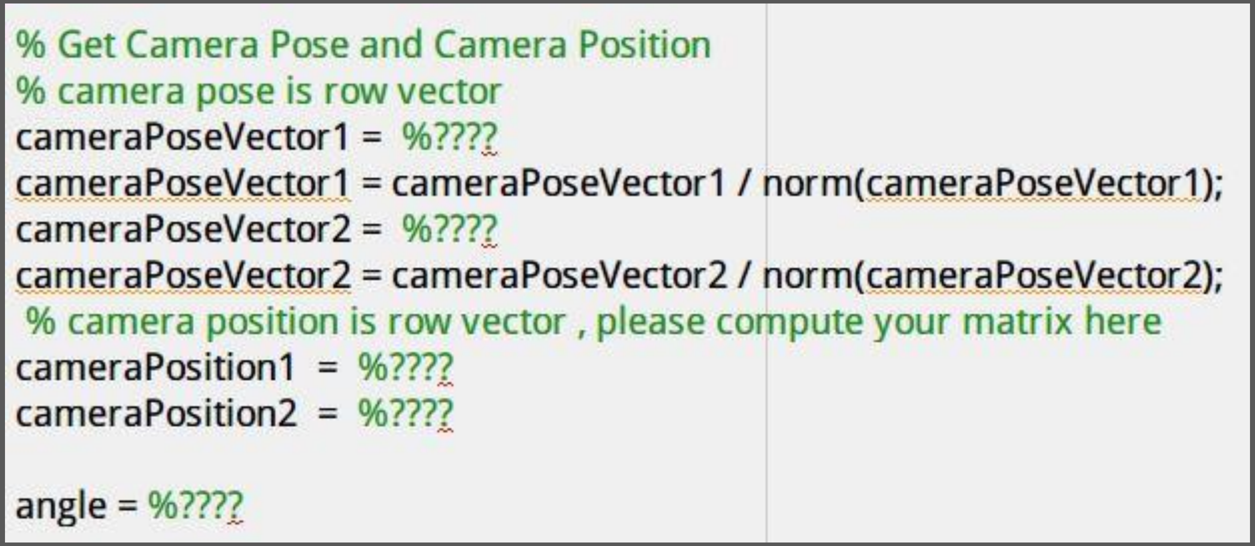
# Part1-D

Visualize result



# Part1-D

- You should complete this part



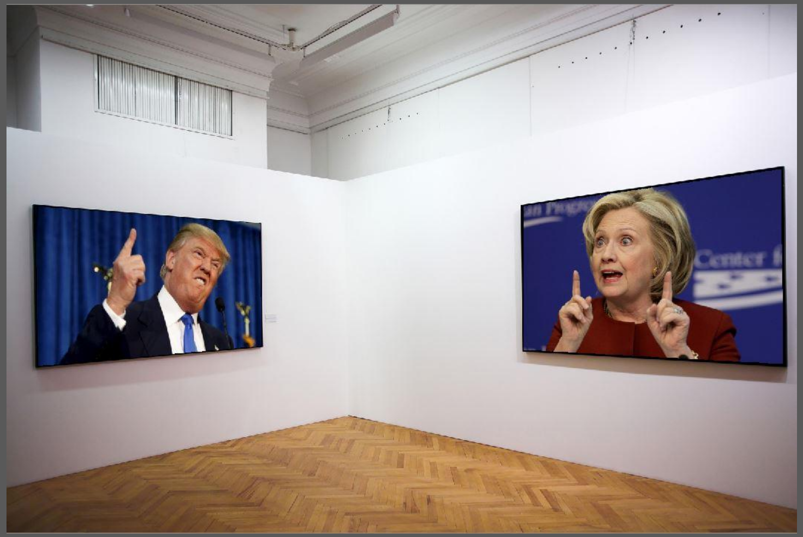
```
% Get Camera Pose and Camera Position
% camera pose is row vector
cameraPoseVector1 =  %????
cameraPoseVector1 = cameraPoseVector1 / norm(cameraPoseVector1);
cameraPoseVector2 =  %????
cameraPoseVector2 = cameraPoseVector2 / norm(cameraPoseVector2);
 % camera position is row vector , please compute your matrix here
cameraPosition1  =  %????
cameraPosition2  =  %????

angle = %????
```

# Part2-B

1.Compute Homography matrix

2.Use bilinear interpolation to do backward warping

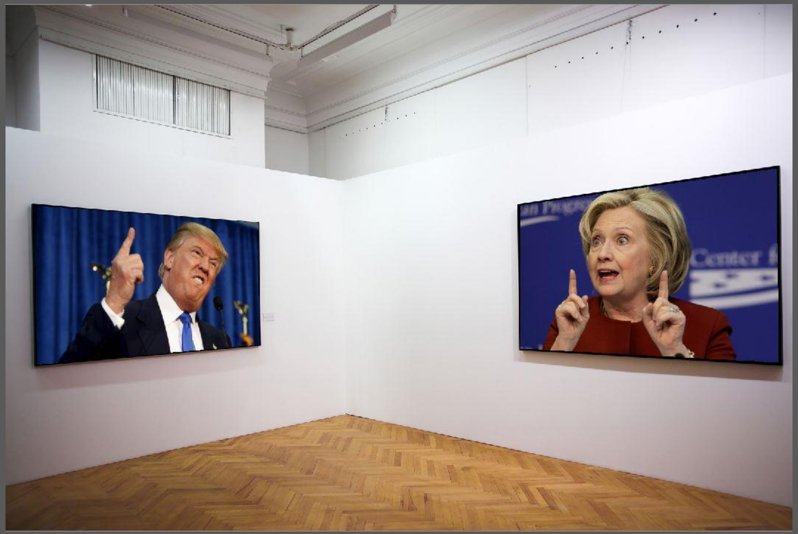


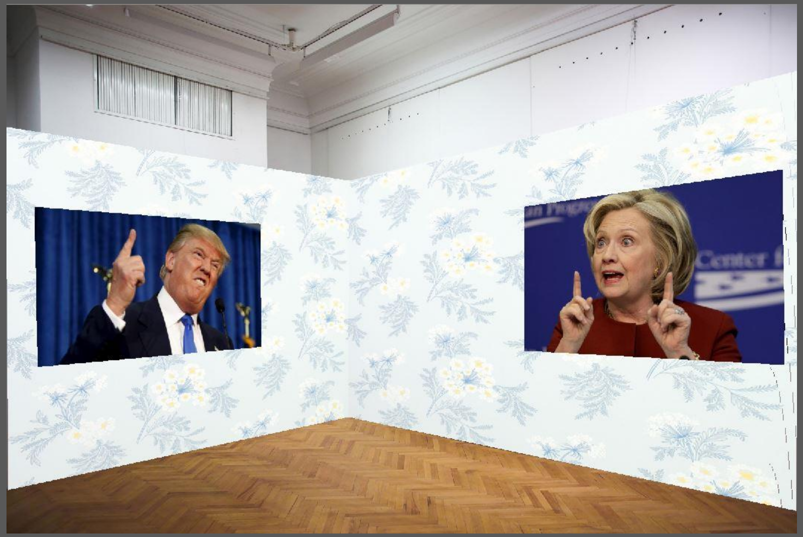
# Part2-C

1.Find your wall texture

2.Compute Homography matrix

3.Use bilinear interpolation to do forward warping





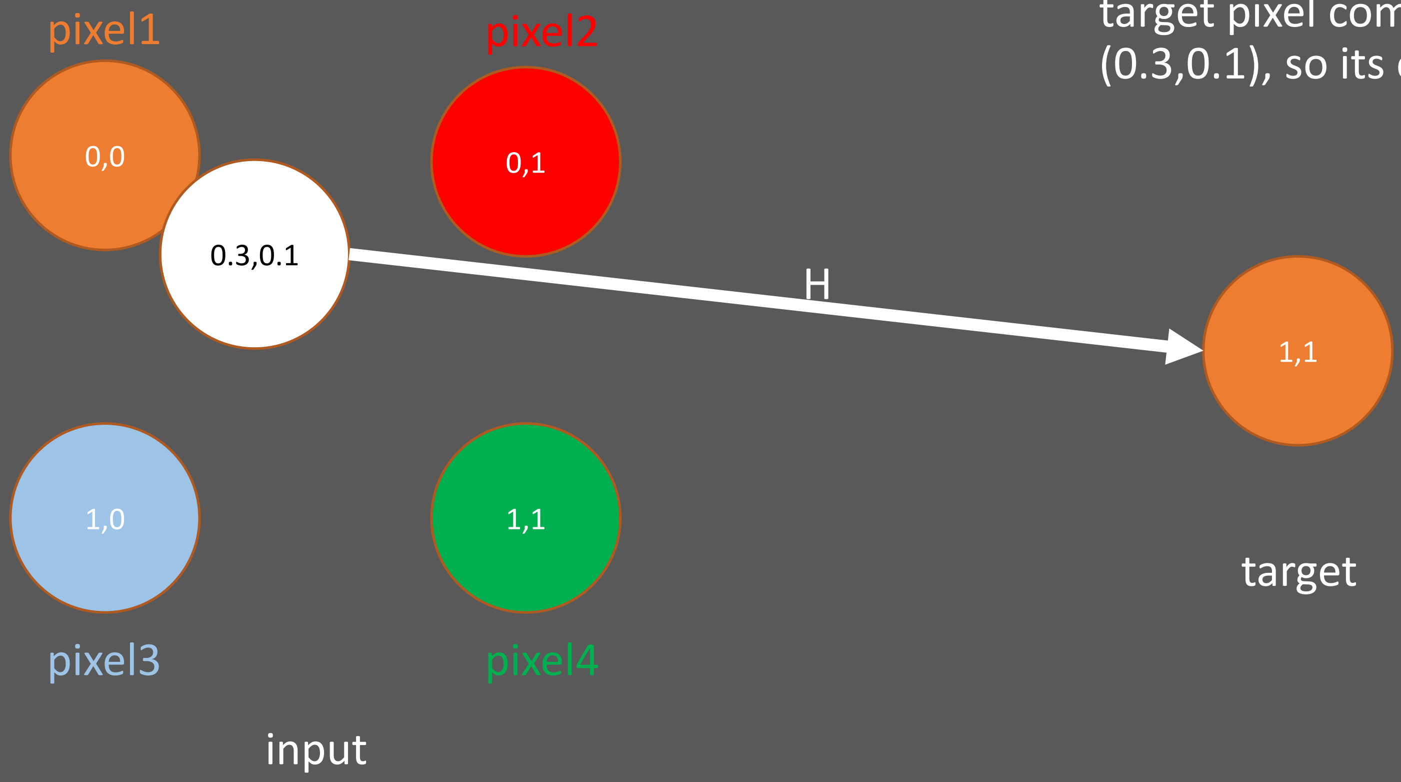
# Part2-B

## Nearest neighbor backward warping

Search what pixels produce you.

target pixel comes from input image location (0.3,0.1), so its color is the nearest one (0,0)

Ex.



# Part2-C

## Nearest neighbor forward warping

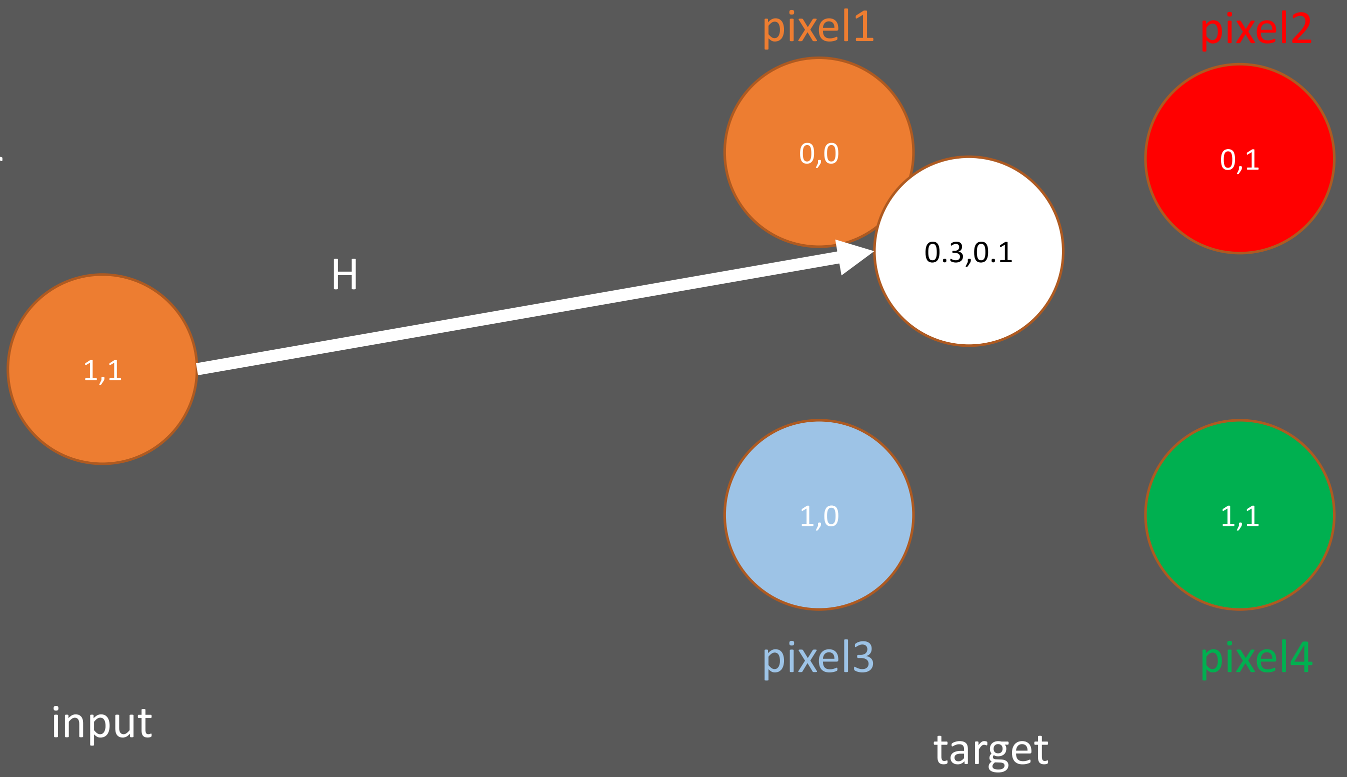
You need to find pixels you offer color to.

Input pixel transform to the location (0.3,0.1)

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

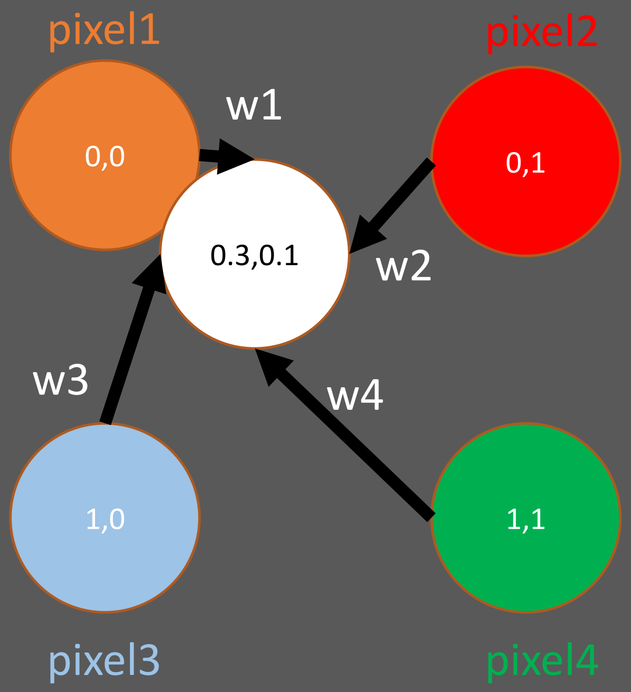
fill with input pixel color

Ex.



# 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


pixel1
0,0
w1
pixel2
0,1
0.3,0.1
w2
w3
w4
1,0
1,1
pixel3
pixel4