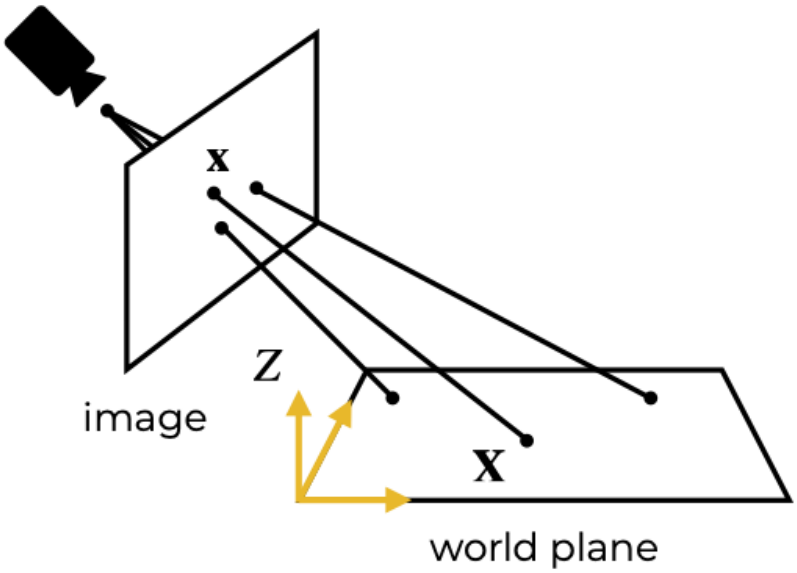
DUCKIETOWN

02 - Homographies

The pinhole camera model provides a many-to-one mapping between points in the scene and their corresponding projection onto the image plane. With knowledge of the intrinsic and extrinsic parameters that specify the camera matrix, we can relate points in the image to their corresponding rays in the scene. However, we don't know where along this ray the corresponding scene point lies.

One way of resolving some of this ambiguity is to use prior knowledge of the environment geometry as a means of adding further structure to the projetion model. In the case of Duckietown, we will use the fact that the environment is planar. In other settings, including self-driving vehicles, while the world may not be globally planar, we can treat it as locally planar around the robot (e.g., imagine the tangent plane on the curved surface that the robot is driving on). We can then exploit this planarity to constrain the projection model.



A homography provides an invertible transformation between the homogeneous coordinates of points in two planes.

Consider the scenario above in which the world is planar and consists of a collection of points $\mathbf{X}_i,\ i \in \{1, 2, \ldots, n\}$, expressed in homogeneous coordinates. These points project to points $\mathbf{x}_i$ in an image of the scene. We can relate each point-pair according to the camera projection matrix

$$\mathbf{x}_i = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} [\,\mathbf{R} \mid \mathbf{t}\,] \mathbf{X}_i$$

where $\mathbf{R}$ and $\mathbf{t}$ are the rotation matrix and translation vector, respectively, that define the transformation from the world frame to the camera frame. Since we are free to define the world frame as we see fit, let's define it such that the world plane lies in the $x - y$ plane with the z-axis orthogonal and pointing up. We can then write the projection operation as follows:

$$\mathbf{x}_i = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{bmatrix} \quad \text{since } Z_i = 0 \tag{2}$$

$$= \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & t_x \\ R_{21} & R_{22} & t_y \\ R_{31} & R_{32} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{4}$$

The matrix H defines a projective transformation and is referred to as a *homography*. The matrix is full-rank and defines an invertible mapping between points in the world and their corresponding projection in the image, i.e.,

$$\mathbf{x}_i = H \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} = H^{-1} \mathbf{x}_i \tag{6}$$

While the homography matrix has 9 elements, we can multiply H by any non-zero constant without affecting the projection. To see this, supose that we have the following transformation involving non-homogeneous image coordinates $[x_i \; y_i]^\top$

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = H \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

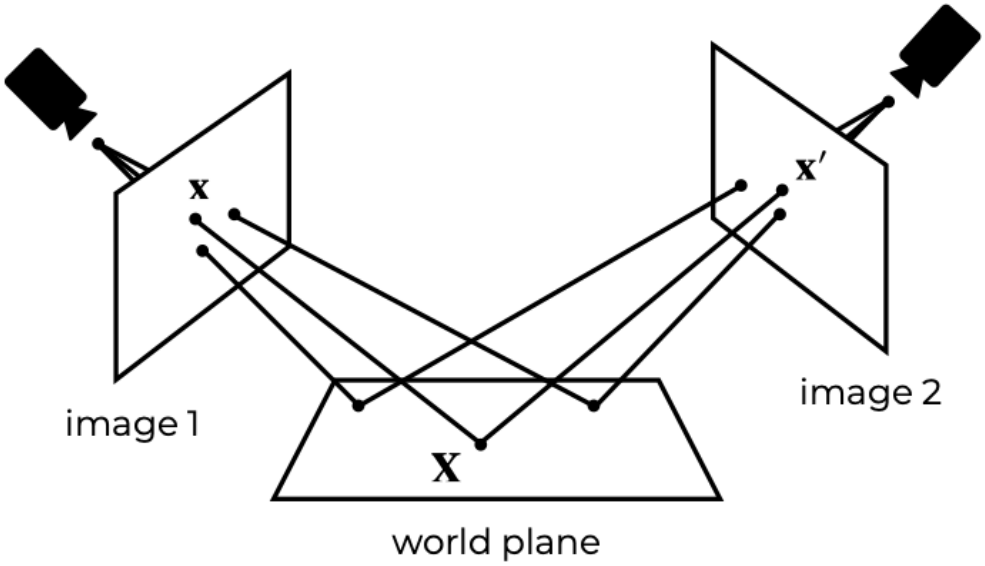
Multiplying H by a non-zero constant λ yields

$$\begin{bmatrix} \lambda x_i \\ \lambda y_i \\ \lambda \end{bmatrix} = (\lambda H) \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

Since these are homogeneous coordinates, the result is the same non-homogeneous coordinates $[x_i \ y_i]^\top$ for the projected point. Thus, while the homography matrix consists of 9 values, it only has 8 degrees-of-freedom.

Example: Exploring a Homography

Homographies define a projection from one plane to another. Above, we described this in the context of a projection H from points that lie on a plane in the world to the image plane for a particular camera pose. The homography provides a one-to-one mapping of points in the world plane to their corresponding projection in the image. If we were to move the camera to a different pose, there would similarly be a different homography $\bar{H}$ relating the image to the ground plane.



A visualization of a simple pinhole camera.

We can compose these homographies to get a homography that relates the homogeneous coordinates of points in one image to their corresponding coordinates in the second image.

$$\begin{aligned} \mathbf{x}_i &= H\mathbf{X}_i && (7) \\ &= H\left(\bar{H}^{-1}\mathbf{x}'_i\right) && (8) \\ &= \left(H\bar{H}^{-1}\right)\mathbf{x}'_i && (9) \\ &= \check{H}\mathbf{x}'_i && (10) \end{aligned}$$

where $\check{H} = H\bar{H}^{-1}$ is the homography between the two images that is induced by the ground plane.

Two images of a hallway on the Oxford campus taken from different viewpoints. The planar ground induces a homography $\check{H}$ between the two images.

In this example, you will explore the use of a homography to relate images of the same scene taken from different poses. While the scene is not planar, there are planes in the

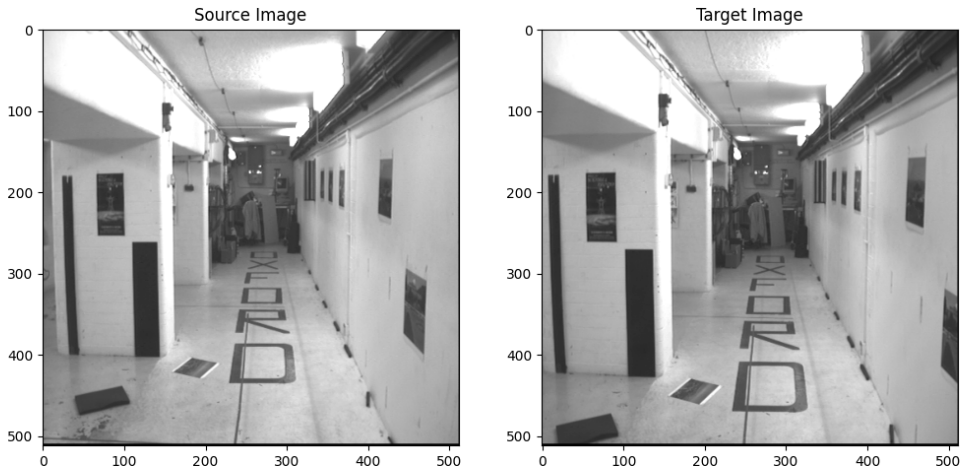
scene that can be used to induce a homography between the two views. In the example below, the floor (ground) plane was used to estimate the homography. In this exercise, you will explore the validity of this homography and see whether it provides a valid transformation between points that don't lie on the ground.

Do you think that the transformation will be valid for points that lie on the walls? Why/why not? (Specifically, pay close attention to the right wall, closer to the camera.)

In [ ]:
```
### Run this cell to import relevant modules
%matplotlib widget

import matplotlib
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

In [ ]:
```
# TODO: Running this cell will bring up a figure displaying the two above im
#.      the projected point in the right image will be rendered according to
#.      Compare the accuracy of the correspondences for points that lie on t
imgl = cv2.imread('../../assets/images/homography/bt.000.png', 0)
imgr = cv2.imread('../../assets/images/homography/bt.002.png', 0)

H = np.array([[0.907503833504229, -0.116496578881938, 30.8471918181923],[0.0

fig = plt.figure(figsize = (12,6))
ax1 = fig.add_subplot(1,2,1)
ax1.imshow(imgl,cmap = 'gray')
ax1.set_title('Source Image')
ax2 = fig.add_subplot(1,2,2)
ax2.imshow(imgr,cmap = 'gray')
ax2.set_title('Target Image')

def onclick(event):
    # Apply the homography
    x = np.array([[event.xdata, event.ydata, 1]]).transpose()
    xprime = H.dot(x)
    xprime = xprime/xprime[2]

    # Visualize the selected and projected points
    ax1.plot(x[0], x[1], 'rx')
    ax2.plot(xprime[0], xprime[1], 'rx')

cid = fig.canvas.mpl_connect('button_press_event', onclick)
```

Figure



You can now move to the camera calibration tutorial.