```python
#The code below is intended to find the homography matrix after feature matching
#The code is adapted from the image alignment tutorial
#The website is as follows: https://www.pyimagesearch.com/2020/08/31/image-alignment-and-registration-with-opencv/

#Import libraries
import numpy as np
import imutils
import cv2
```

```python
#A helper function that finds the homography matrix
#Input: the image/camera input and the template/desired position
#Output: the homography matrix between the two images

def homography_matrix(image, template, maxFeatures=500, keepPercent=0.2,debug=False):

# convert both the input image and template to grayscale
    imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    templateGray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)

# use ORB to detect keypoints and extract (binary) local invariant features
    orb = cv2.ORB_create(maxFeatures)
    (kpsA, descsA) = orb.detectAndCompute(imageGray, None)
    (kpsB, descsB) = orb.detectAndCompute(templateGray, None)

# match the features
    method = cv2.DESCRIPTOR_MATCHER_BRUTEFORCE_HAMMING
    matcher = cv2.DescriptorMatcher_create(method)
    matches = matcher.match(descsA, descsB, None)

# sort the matches by their distance
# the smaller the distance, the "more similar" the features are)

    matches = sorted(matches, key=lambda x:x.distance)
# keep only the top matches

    keep = int(len(matches) * keepPercent)
    matches = matches[:keep]

# allocate memory for the keypoints (x, y)-coordinates from the
# top matches -- we'll use these coordinates to compute the homography matrix

    ptsA = np.zeros((len(matches), 2), dtype="float")
    ptsB = np.zeros((len(matches), 2), dtype="float")

    # loop over the top matches

    for (i, m) in enumerate(matches):

        # indicate that the two keypoints in the respective images
        # map to each other

        ptsA[i] = kpsA[m.queryIdx].pt
        ptsB[i] = kpsB[m.trainIdx].pt

#computes the homography matrix between two sets of matched points

    (H, mask) = cv2.findHomography(ptsA, ptsB, method=cv2.RANSAC)
    return H
```