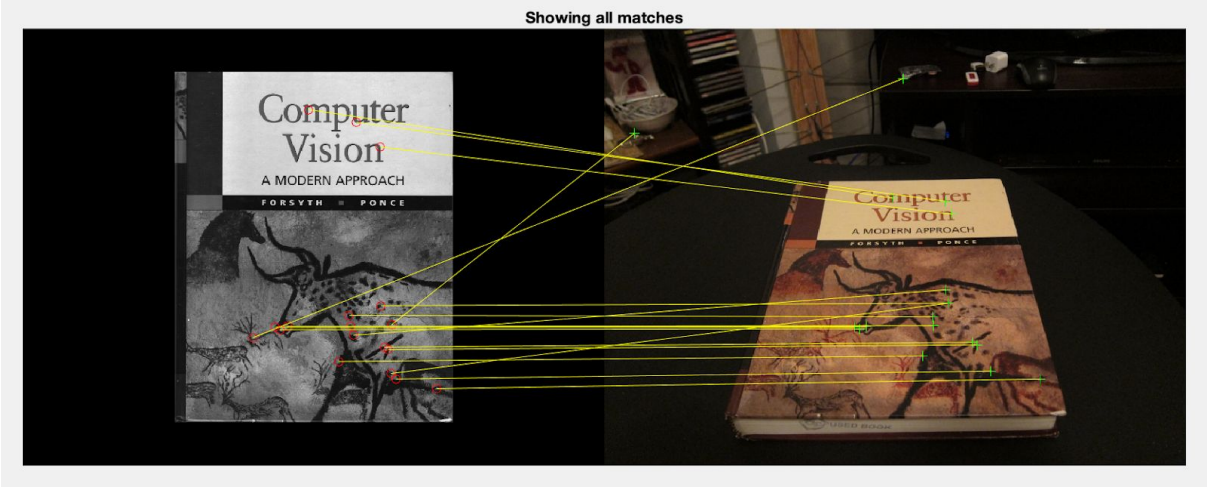
Project 4

4.1

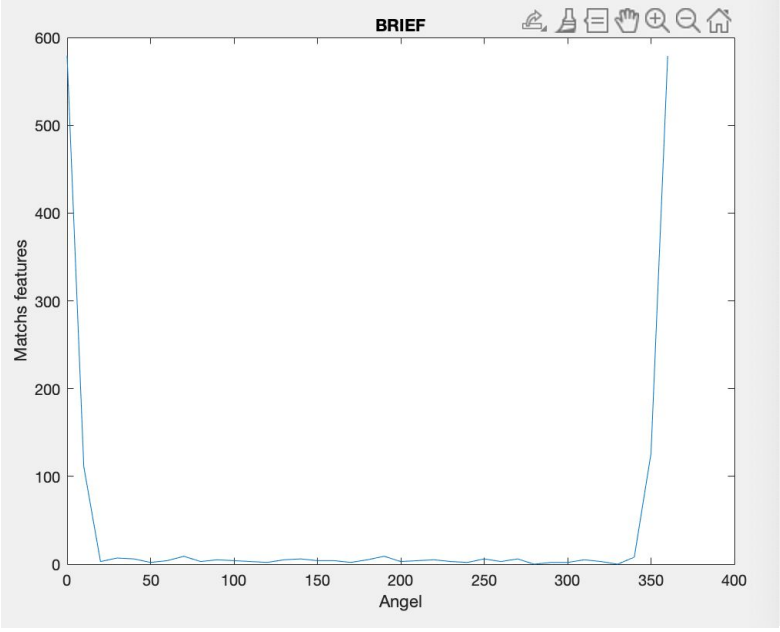
To complete matchPics.m,

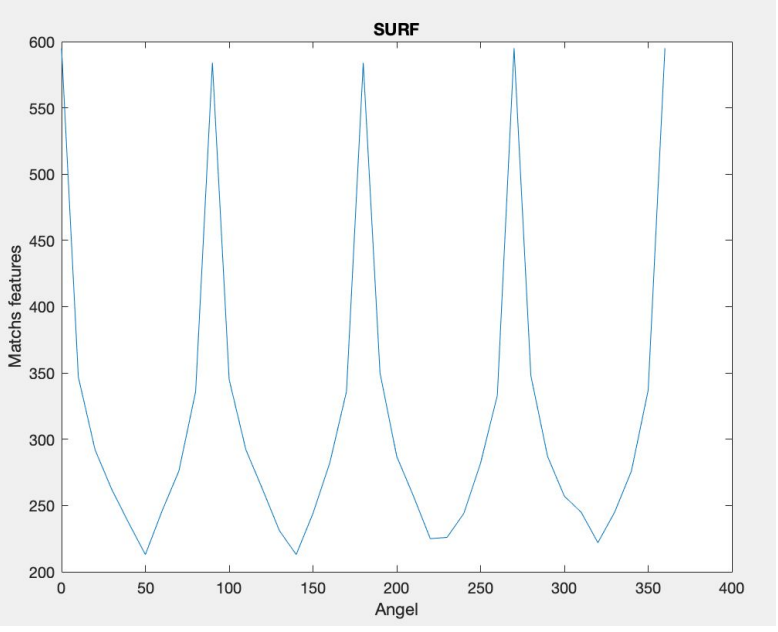
First I use rgb2gray(image) to convert images to grayscale. Then use detectFASTFeature(image) to compute the features and use computeBrief(image , feature.Location) to build descriptors. And set ratio = 0.7 and threshold = 10.0 in matchFeatures().



4.2

Visualize the feature matching result at three different orientations:



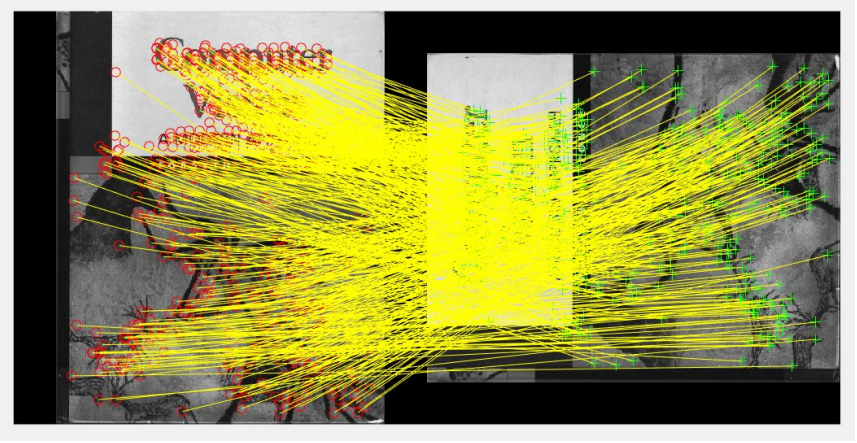


Why you think the BRIEF descriptor behaves this way:
We can see that BRIEF can’t complete the image match features unless the rotation = 0 degree and 360 degree. This should because BRIEF is rotation varitan.It can only match when both images have the same rotation. So that it can not get a good match at different orientations.

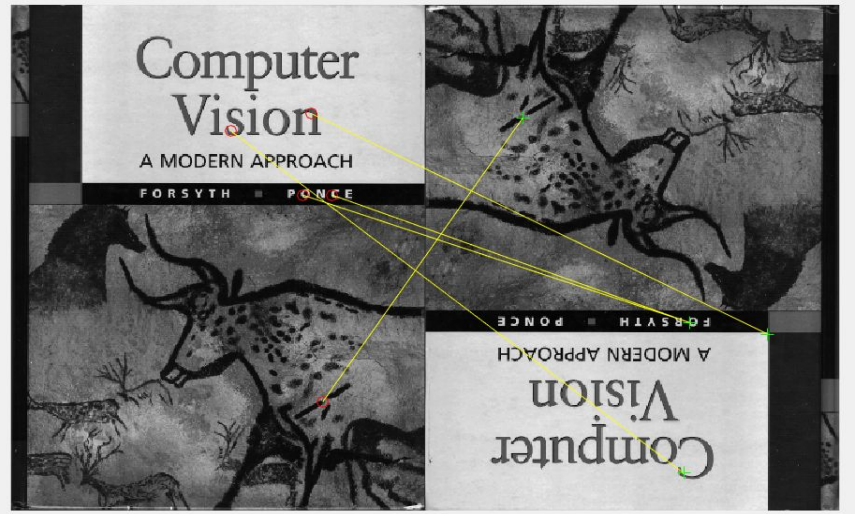
The plot changed significantly in SURF. Compared with BRIEF, SURF gets a better performance. It can get a good match when rotation = 0 ; 90 ; 180 ; 270 ; 360.

When rotation = 90 degree:
BRIEF

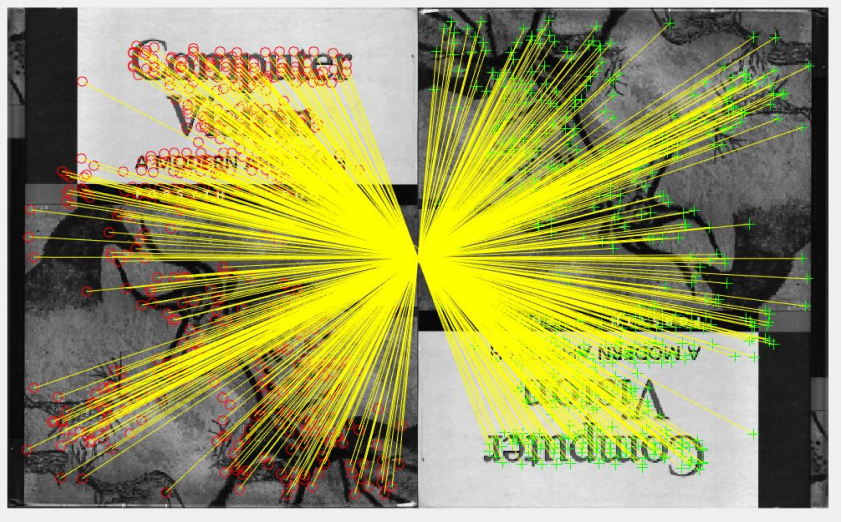
SURF



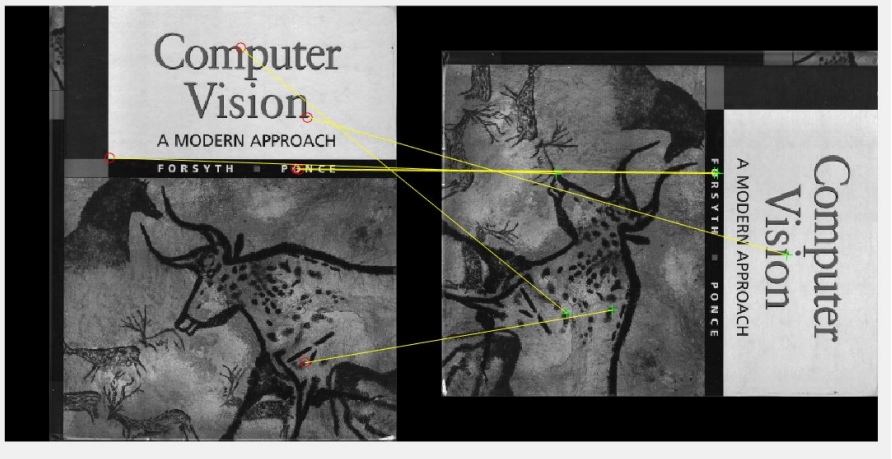
When rotation = 180 degree:
BRIEF



SURF



When rotation = 270 degree:
BRIEF



SURF

4.3

In this section, I use SVD to estimate the homography transformation, get the eigenvectors. And re-write the matrix with the follow rules:

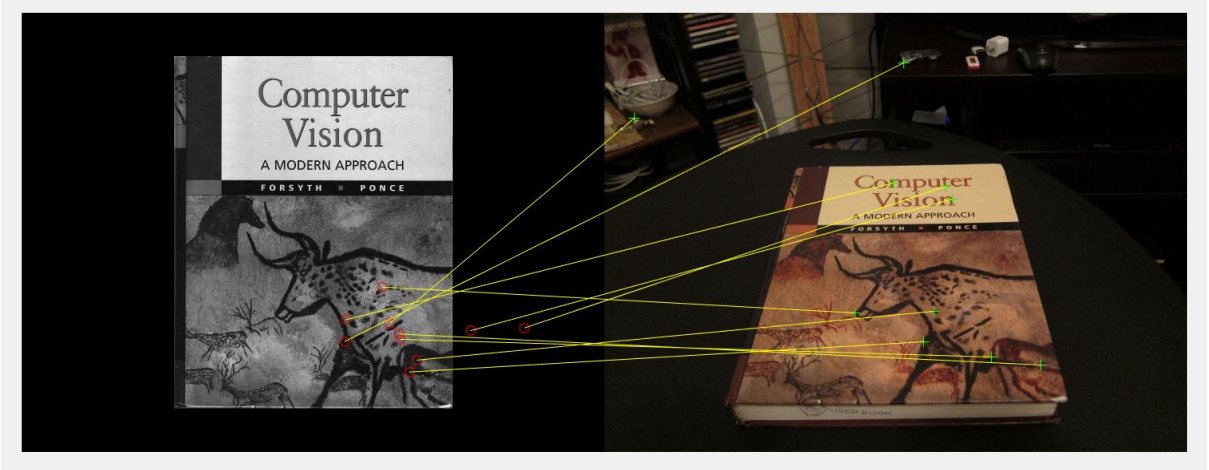
Re-write in matrix form:

$$\mathbf{A}_i \boldsymbol{h} = \mathbf{0}$$

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

$$\boldsymbol{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Visualize the 10 random points between the origin image and homography transform image.



4.4

To compute the normalize, we need to calculate the mean of both points as centroids. Then shift the origin of the points to the centroid by using norm points - centroids. Finally, sqrt(2) / average to make all points have an sqrt(2) average distance to origin point.

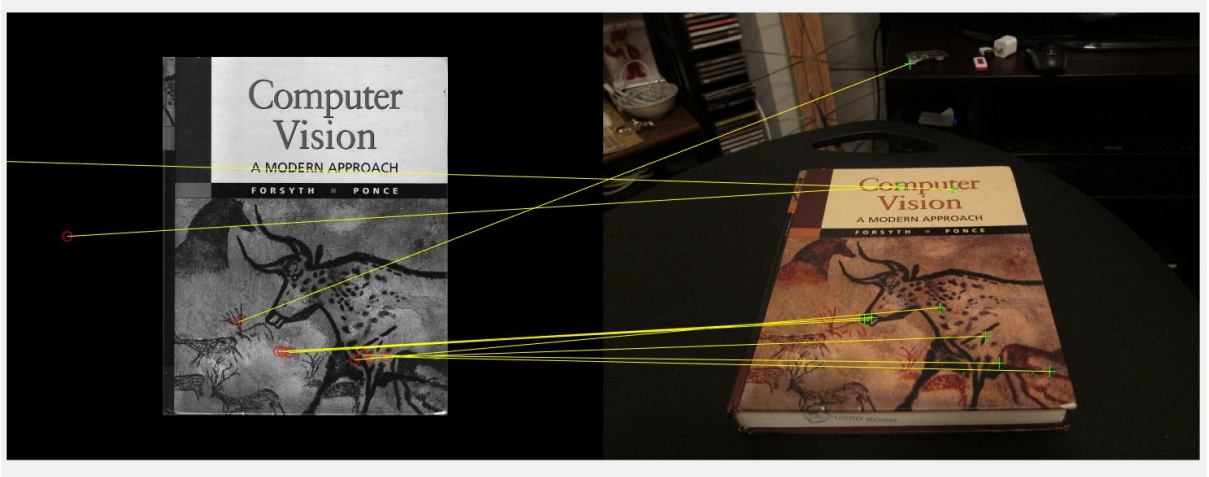
Use the follow matrix to get the similarity transform:

[norm , 0 , -norm * centroid ; 0 , norm, -norm * centroid ; 0 , 0 , 1]

```
%% similarity transform 1
T1 = [x1_norm, 0, -x1_norm*centroid1(1);
      0, x1_norm, -x1_norm*centroid1(2);
      0,       0, 1];
```

```
%% similarity transform 2
T2 = [x2_norm, 0, -x2_norm*centroid2(1);
      0, x2_norm, -x2_norm*centroid2(2);
      0,       0, 1];
```

Visualize the 10 random points between the origin image and homography transform image.



4.5
Follow the algorithm:

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute H using DLT
  3. Count inliers
  4. Keep H if largest number of inliers
- Recompute H using all inliers

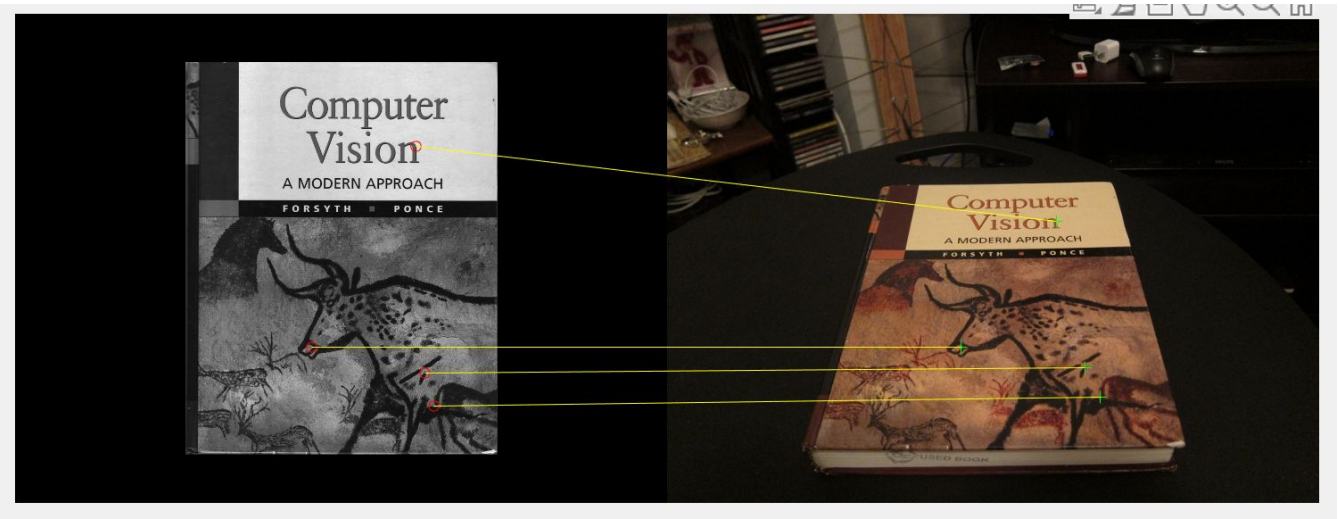
In this section, I set 10000 iterations which should be enough for the largest number of inliers.
the 4 point-pairs (that produced the most number of inliers):

Points in image 1: (X , Y)		Points in image 2: (X , Y)	
330	398	548	440
221	320	414	373
220	306	413	362
133	324	321	377

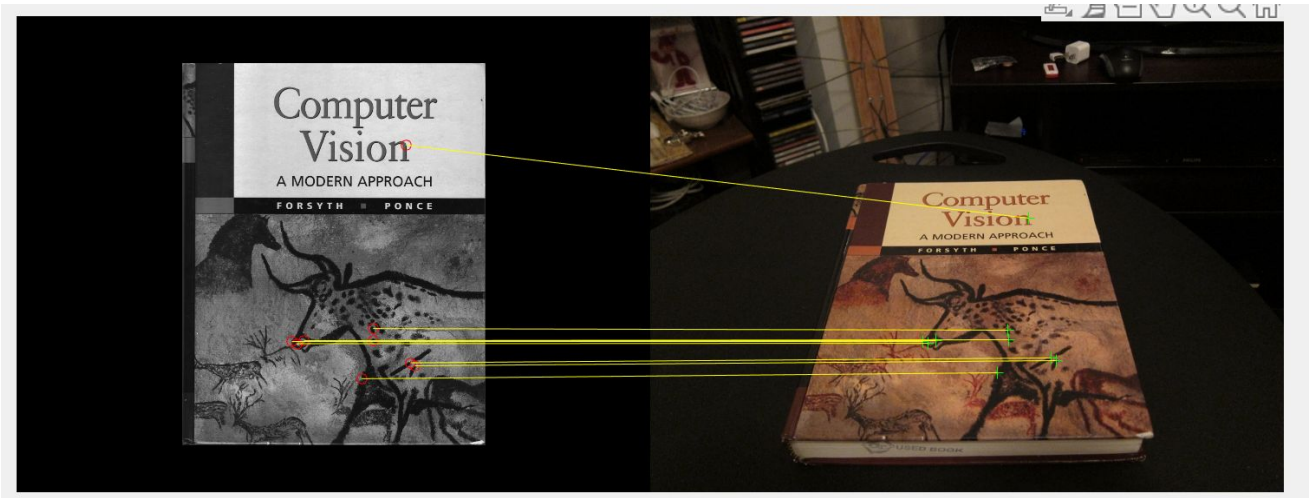
The inlier vector
ans =

0 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1

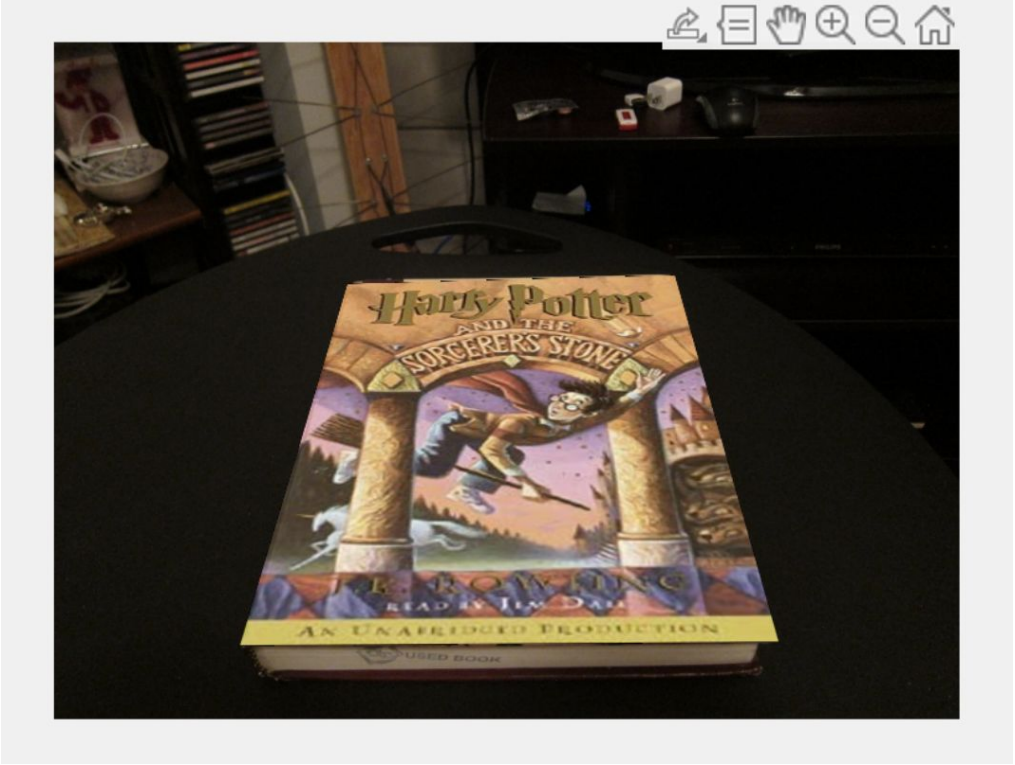
Visualize the 4 point-pairs (that produced the most number of inliers) between the origin image and homography transform image.



Visualize the inlier matches that were selected by the RANSAC algorithm.



4.6
Visualize the result:



In compositeH(H2to1, template, img), I use warpH() function to get the Warp. Then file the warp template with the image when the warp mask equals 0.
5.
To create my Augmented Reality application, first we need to know the length of the video. Then use the short video length as iterations. We know the video is combined with a lot of images, thus, we just need to separate the video as single images and repeat 4.6(HarryPotterize.m) many times. Finally we just need to write the images as video.
I crop the images of video: (50,215) to (310,425), and I create an empty folder, put my new images. Write a new video(ar.avi).
Here is some images sample:


Multiple Vie
Geometr
in computer vis
Richard Hartley and Andrew Zis


SECOND EDITION
Multiple View
Geometry
in computer vision
Richard Hartley and Andrew Zisserman