In [1]:
```
import cv2
import numpy as np
```

In [14]:
```
# Harris corner detection   - need to covernt the image to grayscale and convert to float32
img = cv2.imread("bookimage.jpeg")

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_img = np.float32(gray_img)
# cornerharris(src, blockSize (no. of neighbours), kernel size = )
dst = cv2.cornerHarris(gray_img, blockSize = 2, ksize = 5, k = 0.04)
# dilate to mark the corners (increases the white regions)
dst = cv2.dilate(dst, None)
img[dst>0.1 * dst.max()] = [0,255,0]

cv2.imshow("harris_corner",img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [9]:
```
# SIFT (Scale-Invariant Feature Transform)
img = cv2.imread("bookimage.jpeg", 0)

sift =cv2.xfeatures2d.SIFT_create()
keypoints, descriptors = sift.detectAndCompute(img, None)
#cv.drawKeypoints(image, keypoints, outImage, color, flags] )

img = cv2.drawKeypoints(img, keypoints, None)

cv2.imshow("Image2", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [23]:
```
# shi-Tomasi corner detection for tracking
import cv2
import numpy as np

img = cv2.imread("bookimage.jpeg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

corners = cv2.goodFeaturesToTrack(gray_img, maxCorners = 100, qualityLevel =0.02, minDistance = 10)
gray_img = np.float32(corners)

for item in corners:
    x,y = item[0]
    cv2.circle(img, (x,y), 4, (0,255,0), -1)

cv2.imshow("good features", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
<ipython-input-23-65cc9248891a>:13: DeprecationWarning: an integer is required (got type numpy.float32).  Implicit conversion to integers using __int__ i
s deprecated, and may be removed in a future version of Python.
  cv2.circle(img, (x,y), 4, (0,255,0), -1)
```

In [5]:
```
#Fast algorithm for corner detection
import cv2
import numpy as np

img = cv2.imread("bookimage.jpeg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

fast = cv2.FastFeatureDetector_create()
fast.setNonmaxSuppression(False)

kp = fast.detect(gray_img, None)
kp_img = cv2.drawKeypoints(img, kp, None, color = (0,255,0))


cv2.imshow("fast", kp_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [25]:
```
#ORB (Oriented FAST and Rotated Brief)
import cv2
import numpy as np

img = cv2.imread("bookimage.jpeg")
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

orb = cv2.ORB_create(nfeatures=1000)
kp, des = orb.detectAndCompute(gray_img, None)


kp_img = cv2.drawKeypoints(img, kp, None, color = (0,255,0))


cv2.imshow("ORB", kp_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [26]:
```
# feature matching
import cv2
import numpy as np
from matplotlib import pyplot as plt

img1 = cv2.imread('bookimage.jpeg', 0)   #original image
img2 = cv2.imread('bookimage1.jpeg', 0)   # in scene

# initilize orb
orb = cv2.ORB_create()
# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# matcher takes normType, which is set to cv2.NORM_L2 for SIFT and SURF, cv2.NORM_HAMMING for ORB, FAST and BRIEF
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
# match descriptors
matches = bf.match(des1, des2)
#sort them in order of their distance
matches = sorted(matches, key=lambda x: x.distance)
# draw first 50 matches
match_img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], None)
cv2.imshow('Matches', match_img)
cv2.waitKey()
```

Out[26]:
```
-1
```

In [ ]:
```

```