Building a Star feature detector

SIFT feature detector is good in many cases. However, when we build object recognition systems, we may want to use a different feature detector before we extract features using SIFT. This will give us the flexibility to cascade different blocks to get the best possible performance. So, we will use the **Star feature detector** in this case to see how to do it.

How to do it...

1. Create a new Python file, and import the following packages:

```
import sys

import cv2
import numpy as np
```

2. Define a class to handle all the functions that are related to Star feature detection:

```
class StarFeatureDetector(object):
    def __init__(self):
        self.detector = cv2.xfeatures2d.StarDetector_create()
```

3. Define a function to run the detector on the input image:

```
    def detect(self, img):
        return self.detector.detect(img)
```

4. Load the input image in the `main` function. We will use `table.jpg`:

```
if __name__=='__main__':
    # Load input image -- 'table.jpg'
```

```
input_file = sys.argv[1]
input_img = cv2.imread(input_file)
```

5. Convert the image to grayscale:

```
# Convert to grayscale
img_gray = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
```

6. Detect features using the Star feature detector:

```
# Detect features using Star feature detector
keypoints = StarFeatureDetector().detect(input_img)
```

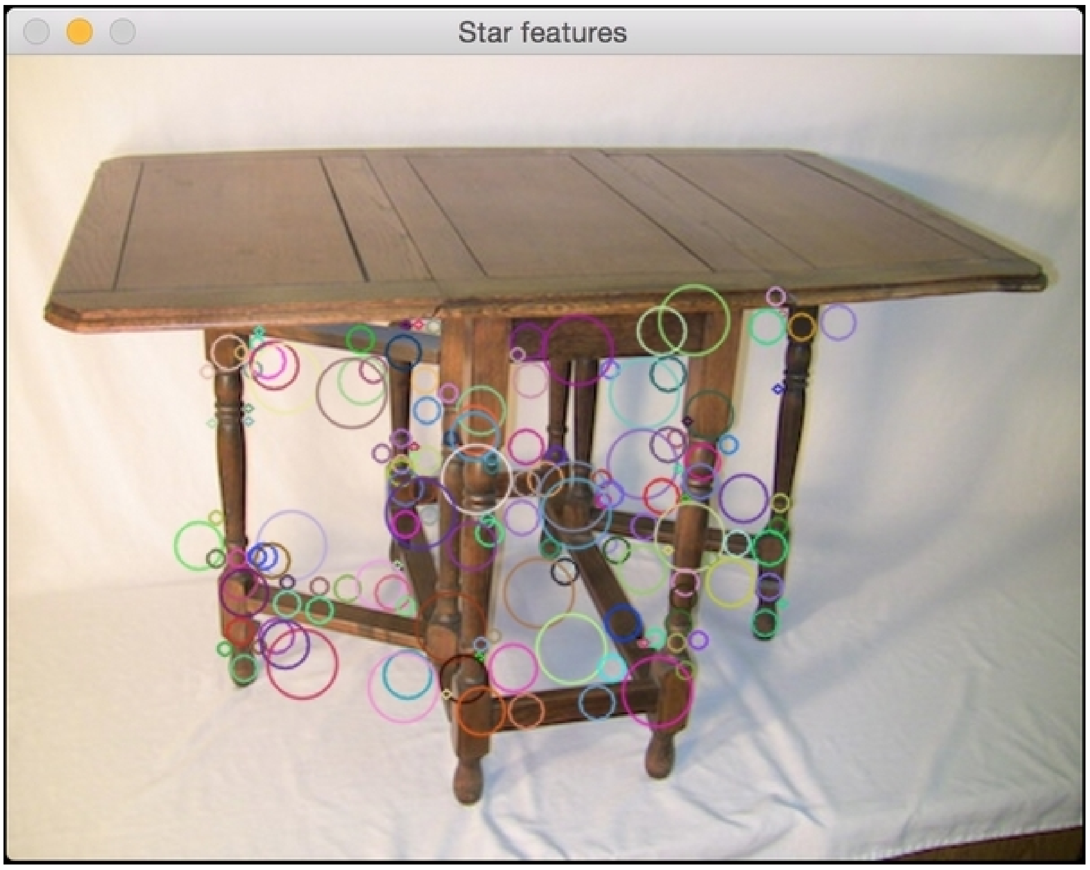
7. Draw keypoints on top of the input image:

```
cv2.drawKeypoints(input_img, keypoints, input_img,
           flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

8. Display the output image:

```
cv2.imshow('Star features', input_img)
cv2.waitKey()
```

9. The full code is given in the `star_detector.py` file that is already provided to you. The output image looks like the following:


Star features