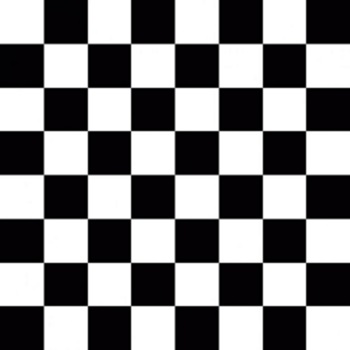
The purpose of detecting corners is to track things like motion, do 3D modeling, and recognize objects, shapes, and characters.

<https://docs.opencv.org/3.0-alpha/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html>



Our goal here is to find all of the corners in this image. I will note that we have some aliasing issues here (jagged-ness in slanted lines), so, if we let it, a lot of corners will be found, and rightly-so. As usual with OpenCV, the hard part is done for us already, and all we need to do is feed in some parameters. Let's start with loading the image and setting some parameters:

import numpy as np

import cv2

img = cv2.imread('opencv-corner-detection-sample.jpg')

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = np.float32(gray)

corners = cv2.goodFeaturesToTrack(gray, 100, 0.01, 10)

corners = np.int0(corners)

So far, we load the image, convert to gray, then to float32. Next, we detect corners with the [goodFeaturesToTrack](https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html" \l "goodfeaturestotrack" \t "blank) function. The parameters here are the image, max corners to detect, quality, and minimum distance between corners. As mentioned before, the aliasing issues we have here will allow for many corners to be found, so we put a limit on it. Next:

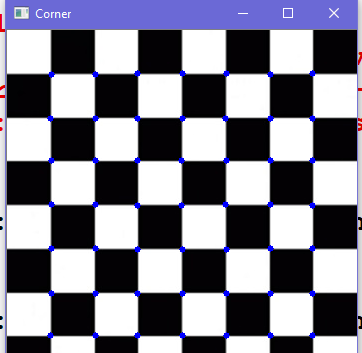
for corner in corners:

x,y = corner.ravel()

cv2.circle(img,(x,y),3,255,-1)

cv2.imshow('Corner',img)

Now we iterate through each corner, making a circle at each point that we think is a corner.



<http://victor-riascos.blogspot.com/2013/01/harris-corner-detector.html>