Tien Li Shen

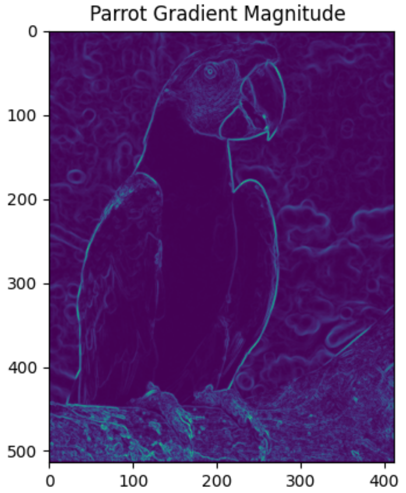
30930512

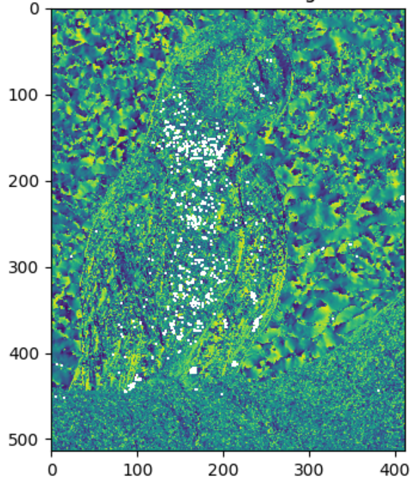
March 1, 2021

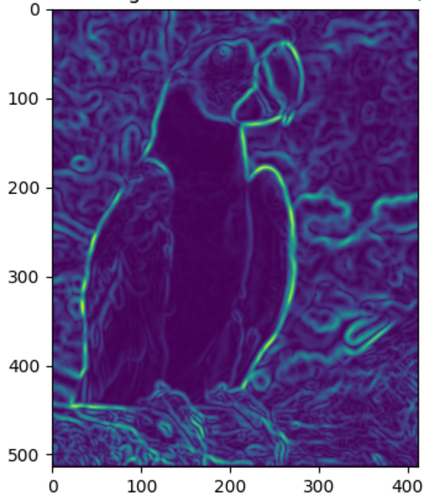
CS 370

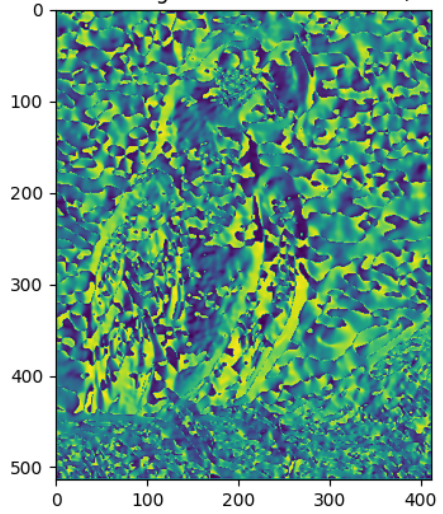
Mini Project 4

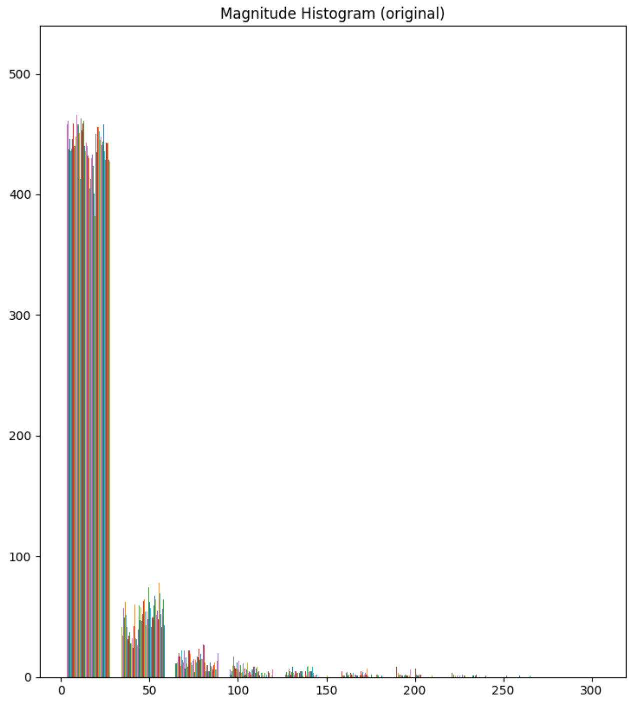
Q1.

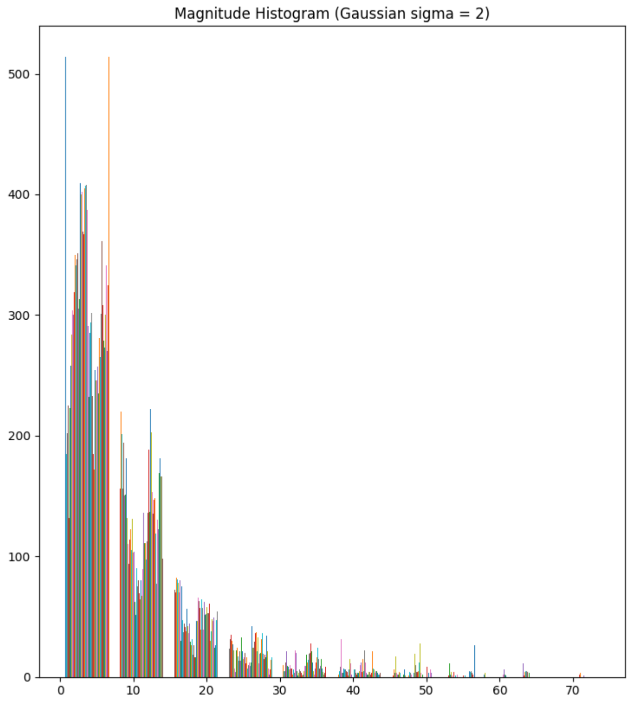

Parrot Gradient Magnitude


Parrot Gradient Angle

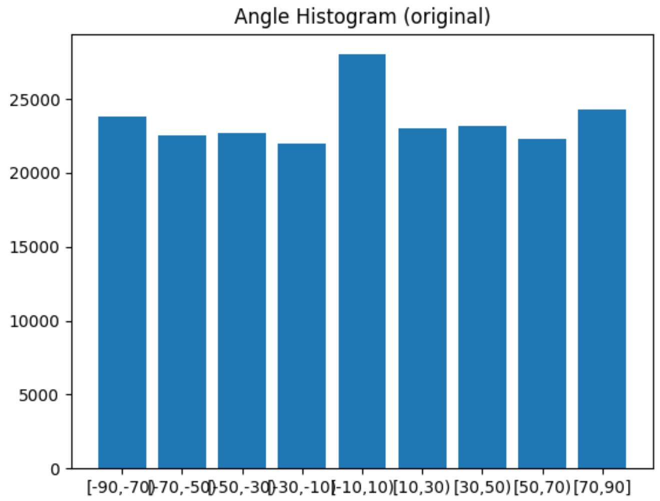

Parrot Gradient Magnitude with Gaussian Filter(simga = 2)

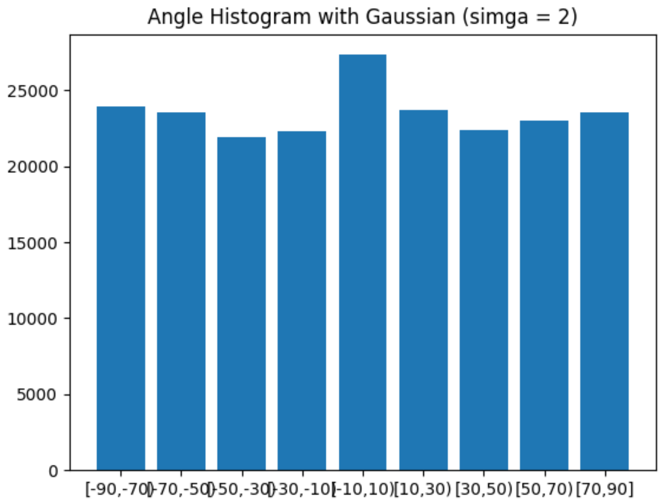

Parrot Gradient Angle with Gaussian Filter(simga = 2)





The gaussian filter created an increase in gradient magnitude.





```
import numpy as np
from skimage.color import rgb2gray
from scipy.ndimage.filters import convolve
from nms import nms
import matplotlib.pyplot as plt
from math import pi
from scipy.ndimage import gaussian_filter

def imageGradient(im):
   f = [-1, 0, 1]

   im_mag = np.zeros(im.shape)
   im_ang = np.zeros(im.shape)
   gx = 0
   gy = 0
   for i in range(1, im.shape[0]-1):
       for j in range(1, im.shape[1]-1):
           gx = np.dot(im[i][j - 1:j + 2], [-1, 0, 1])  # get the gradient in the x
direction
           gy = np.dot([im[i-1][j], im[i][j], im[i+1][j]], [-1, 0, 1]) # get the gradient in
the y direction

           im_mag[i][j] = (gx**2 + gy**2)**(1/2)
           im_ang[i][j] = np.arctan(gy/gx)

   im_ang = np.degrees(im_ang)
   return [im_mag, im_ang]

def place2bin(im_ang):
   count_arr = np.zeros(9)
   # use nested for loop to place each angle into bins
   for row in im_ang:
       for ang in row:
           if ang >= -90 and ang < -70:
               count_arr[0]+=1
           elif ang >= -70 and ang < -50:
               count_arr[1]+=1
           elif ang >= -50 and ang < -30:
               count_arr[2]+=1
           elif ang >= -30 and ang < -10:
               count_arr[3]+=1
           elif ang >= -10 and ang < 10:
               count_arr[4]+=1
           elif ang >= 10 and ang < 30:
               count_arr[5]+=1
           elif ang >= 30 and ang < 50:
               count_arr[6]+=1
           elif ang >= 50 and ang < 70:
               count_arr[7]+=1
           else:
               count_arr[8]+=1
   return count_arr

if __name__ == "__main__":
   im = plt.imread("../data/parrot.jpg")
   im = rgb2gray(np.asarray(im).astype(float))
   # im = gaussian_filter(im, sigma=2)


   im = np.pad(im, ((1, 1), (1, 1)), 'edge')
   im_mag, im_ang = imageGradient(im)
   bin_ang = place2bin(im_ang)
   print(im_ang.shape)
   plt.figure(1)
   plt.title("Parrot Gradient Angle with Gaussian Filter(simga = 2)")
   # plt.title("Parrot Gradient Angle")
   plt.imshow(im_ang, cmap = 'viridis')
   plt.figure(2)
   plt.title("Angle Histogram (original)")
   label = ["[-90,-70)", "[-70,-50)", "[-50,-30)", "[-30,-10)", "[-10,10)", "[10,30)",
"[30,50)", "[50,70)", "[70,90]"]
   plt.bar(x=label, height=bin_ang)
   plt.figure(3)
   plt.subplot(121)
   plt.hist(im_mag)
   plt.title("Magnitude Histogram (original)")
   plt.subplot(122)
   im = gaussian_filter(im, sigma=2)
   im_mag, im_ang = imageGradient(im)
   plt.hist(im_mag)
   plt.title("Magnitude Histogram (Gaussian sigma = 2)")
   plt.show()
```

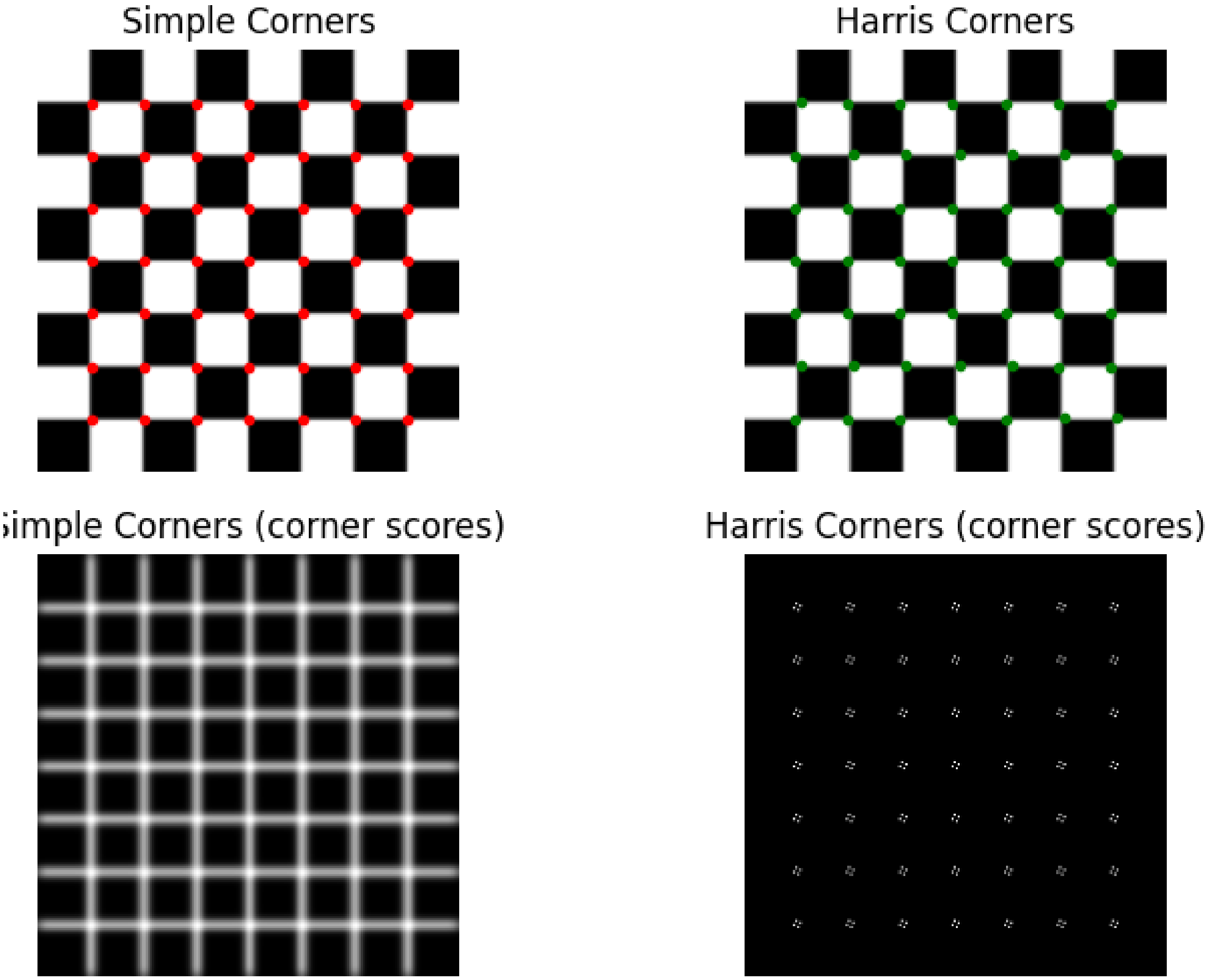
Q2.

The following images show the results from simple and Harris corners for a picture of the checkerboard.

I tweaked the hyperparameters to the following:

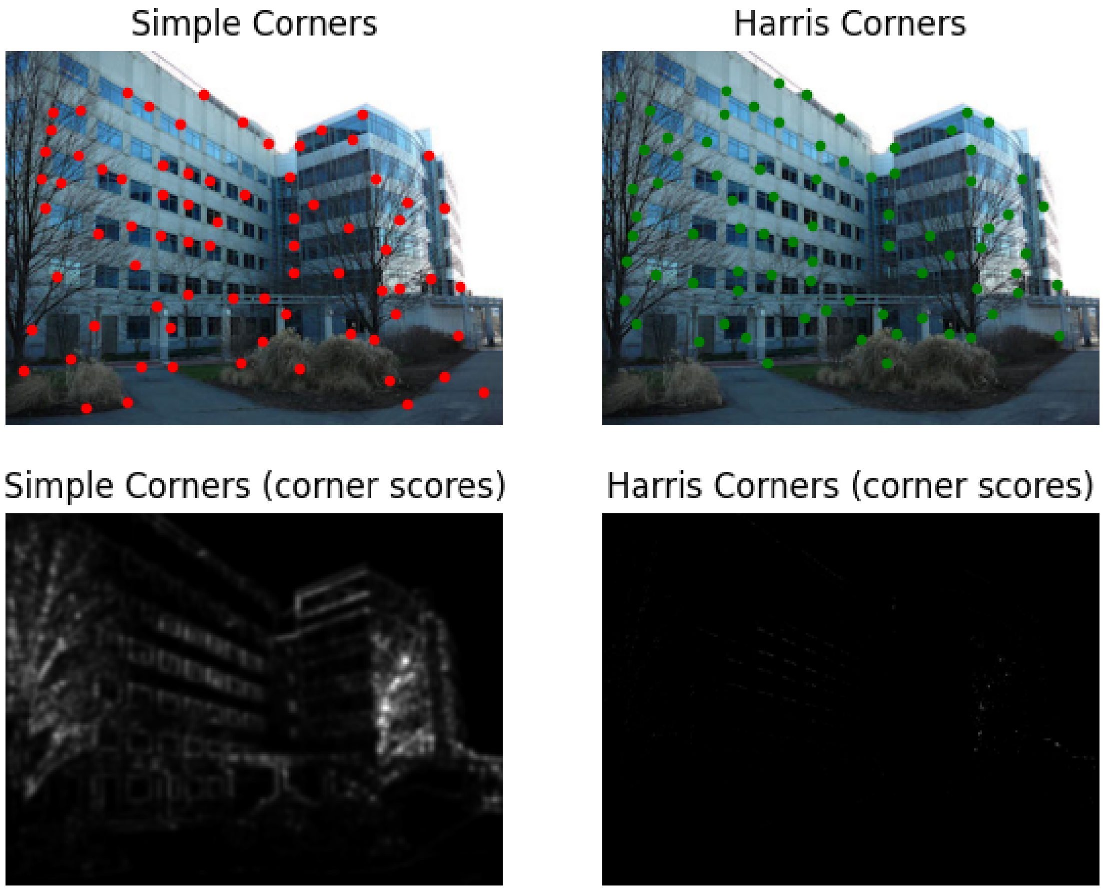
```
cx, cy, cs, corner_score = detectCorners(I, True, 2, 0.05)

cx, cy, cs, corner_score = detectCorners(I, False, 0.5, 0.0005)
```



The following images show the results from simple and Harris corners for polymer-science-umass. I tweaked the hyperparameters to the following:

```
cx, cy, cs, corner_score = detectCorners(I, True, 1.5, 0.005)
cx, cy, cs, corner_score = detectCorners(I, False, 0.5, 0.0005)
```



Simple Corners

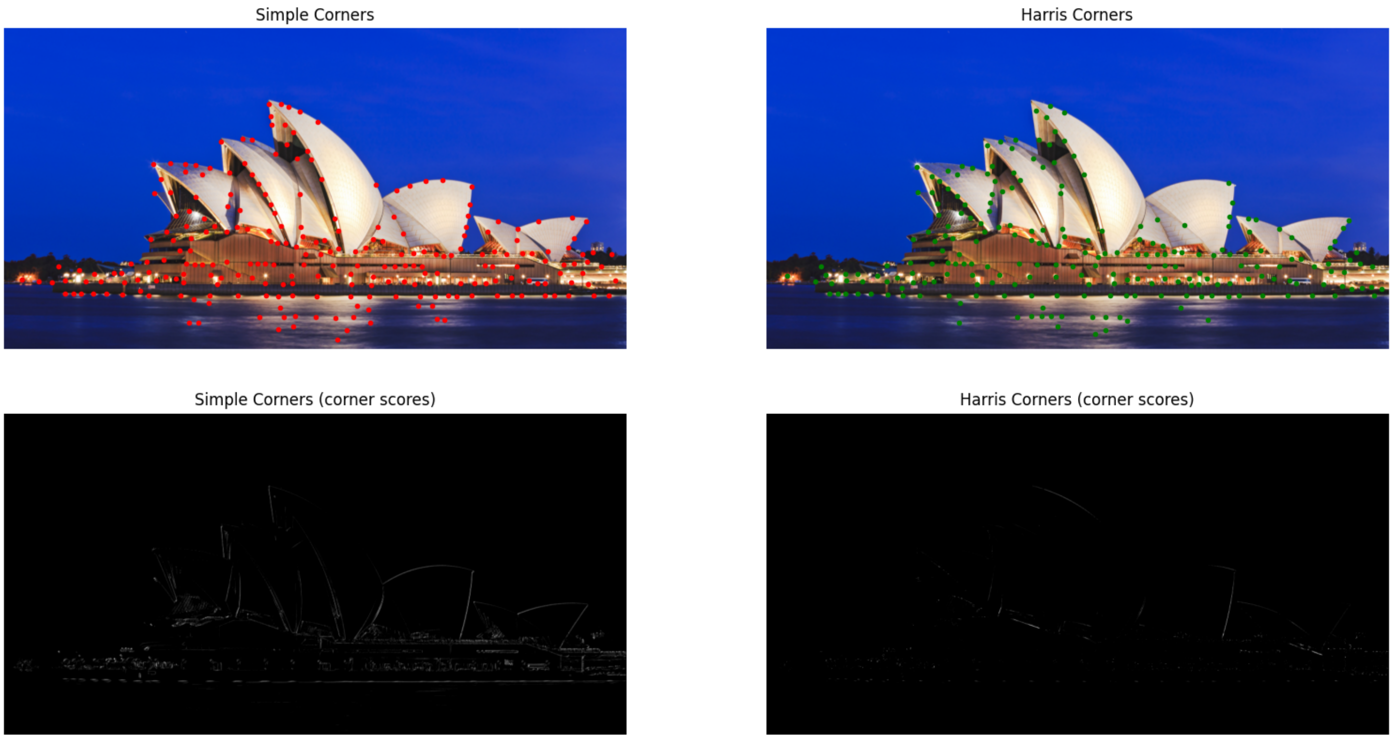
Harris Corners

Simple Corners (corner scores)

Harris Corners (corner scores)

The following images show the results from simple and Harris corners for a picture of the Sydney Opera House. I tweaked the hyperparameters to the following:

```
cx, cy, cs, corner_score = detectCorners(I, True, 0.5, 0.05)
cx, cy, cs, corner_score = detectCorners(I, False, 0.5, 0.0001)
```



Simple Corners

Harris Corners

Simple Corners (corner scores)

Harris Corners (corner scores)

Implementation:

```
def simple_score(I, w):
   corner_score = np.zeros(I.shape)
   # E(u; v) = (I * f(u; v))^2 * Gsigma
   # use a nested for loop to find the gradient
   for i in range(1, I.shape[0]-1):
       for j in range(1, I.shape[1]-1):
           for u in range(-1, 1):
               for v in range(-1, 1):
                   corner_score[i][j] += I[i+u][j+v] - I[i][j]
   corner_score = corner_score**2
   corner_score = gaussian_filter(corner_score, sigma = w)

   return corner_score


#--------------------------------------------------------------------------
#                                    Harris score function (Implement this)
#--------------------------------------------------------------------------
def harris_score(I, w):
   k = 0.04  # as suggested by the assignment pdf
   corner_score = np.zeros(I.shape)
   # get first derivative
   ix, iy = gradient(I)

   # get second order derivative and gaussian filter the second order derivative
   ixx = gaussian_filter(ix ** 2, sigma=w)
   ixy = gaussian_filter(ix * iy, sigma=w)
   iyy = gaussian_filter(iy ** 2, sigma=w)

   # lamda1 * lambd2 = det(M) = ad - bc, and lambda1 + lambda2 = trace(M) = a + d.
Thus you can compute the score as,
   # cornerScore = (ad - bc) - k(a + d)2:
   corner_score = ixx*ixy - iyy*ixy - k*((ixx+iyy)**2)
   return corner_score

def gradient(im):
   gx = np.zeros(im.shape)
   gy = np.zeros(im.shape)
   for i in range(1, im.shape[0]-1):
       for j in range(1, im.shape[1]-1):
           gx[i][j] = np.dot(im[i][j - 1:j + 2], [-1, 0, 1])  # get the gradient in
the x direction
           gy[i][j] = np.dot([im[i-1][j], im[i][j], im[i+1][j]], [-1, 0, 1]) # get the
gradient in the y direction
   return [gx, gy]
```