**Implementation of 3D Vision Techniques**

**Task 1: To implement camera calibration with Chessboard and Circular grid.**

import cv2

import numpy as np

import glob

import matplotlib.pyplot as plt

CHECKERBOARD = (6,9)

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

objpoints = []

imgpoints = []

objp = np.zeros((1, CHECKERBOARD[0]\*CHECKERBOARD[1], 3), np.float32)

objp[0,:,:2] = np.mgrid[0:CHECKERBOARD[0], 0:CHECKERBOARD[1]].T.reshape(-1, 2)

prev\_img\_shape = None

images = glob.glob('./images/\*.jpg')

for fname in images:

    img = cv2.imread(fname)

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

    ret, corners = cv2.findChessboardCorners(gray, CHECKERBOARD, cv2.CALIB\_CB\_ADAPTIVE\_THRESH +

                                             cv2.CALIB\_CB\_FAST\_CHECK + cv2.CALIB\_CB\_NORMALIZE\_IMAGE)

    if ret == True:

        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray, corners, (11,11), (-1,-1), criteria)

        imgpoints.append(corners2)

        img = cv2.drawChessboardCorners(img, CHECKERBOARD, corners2, ret)

        img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

        plt.figure(figsize=(10, 7))

        plt.imshow(img\_rgb)

        plt.title(f'Checkerboard corners - {fname}')

        plt.axis('off')

        plt.show()

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)

print("Camera matrix: \n", mtx)

print("Distortion coefficients: \n", dist)

print("Rotation vectors: \n", rvecs)

print("Translation vectors: \n", tvecs)

import numpy as np

import cv2 as cv

import glob

import matplotlib.pyplot as plt

criteria = (cv.TERM\_CRITERIA\_EPS + cv.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

obj3d = np.zeros((44, 3), np.float32)

a = [0, 36, 72, 108, 144, 180, 216, 252, 288, 324, 360]

b = [0, 72, 144, 216, 36, 108, 180, 252]

for i in range(0, 44):

obj3d[i] = (a[i // 4], (b[i % 8]), 0)

obj\_points = []

img\_points = []

images = glob.glob('./cimages/\*.png')

for f in images:

img = cv.imread(f)

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

ret, corners = cv.findCirclesGrid(gray, (4, 11), None, flags=cv.CALIB\_CB\_ASYMMETRIC\_GRID)

if ret == True:

obj\_points.append(obj3d)

corners2 = cv.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)

img\_points.append(corners2)

cv.drawChessboardCorners(img, (4, 11), corners2, ret)

img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)

plt.figure(figsize=(10, 7))

plt.imshow(img\_rgb)

plt.title('Detected Circular Grid Corners')

plt.axis('off')

plt.show()

ret, camera\_mat, distortion, rotation\_vecs, translation\_vecs = cv.calibrateCamera(

obj\_points, img\_points, gray.shape[::-1], None, None)

print("Error in projection : \n", ret)

print("\nCamera matrix : \n", camera\_mat)

print("\nDistortion coefficients : \n", distortion)

print("\nRotation vector : \n", rotation\_vecs)

print("\nTranslation vector : \n", translation\_vecs)

**Task 2: To detect crack in the image using “shape from shading” technique.**

import cv2

import numpy as np

import matplotlib.pyplot as plt

img = cv2.imread('c1.png', cv2.IMREAD\_GRAYSCALE)

img\_blur = cv2.GaussianBlur(img, (5, 5), 0)

sobelx = cv2.Sobel(img\_blur, cv2.CV\_64F, 1, 0, ksize=5)

sobely = cv2.Sobel(img\_blur, cv2.CV\_64F, 0, 1, ksize=5)

gradient\_magnitude = np.sqrt(sobelx\*\*2 + sobely\*\*2)

gradient\_magnitude = cv2.normalize(gradient\_magnitude, None, 0, 255, cv2.NORM\_MINMAX)

gradient\_magnitude = np.uint8(gradient\_magnitude)

\_, cracks = cv2.threshold(gradient\_magnitude, 50, 255, cv2.THRESH\_BINARY)

h, w = cracks.shape

center = (w // 2, h // 2)

radius = min(center[0], center[1]) - 15

mask = np.zeros\_like(cracks)

cv2.circle(mask, center, radius, 255, thickness=-1)

cracks\_masked = cv2.bitwise\_and(cracks, mask)

kernel = np.ones((3, 3), np.uint8)

cracks\_refined = cv2.morphologyEx(cracks\_masked, cv2.MORPH\_CLOSE, kernel)

cracks\_refined = cv2.morphologyEx(cracks\_refined, cv2.MORPH\_OPEN, kernel)

plt.figure(figsize=(10, 8))

plt.subplot(1, 2, 1)

img\_color = img

plt.title("Original Image")

plt.imshow(cv2.cvtColor(img\_color, cv2.COLOR\_BGR2RGB))

plt.axis('off')

plt.subplot(1, 2, 2)

plt.title("Detected Cracks")

plt.imshow(cracks\_refined, cmap='gray')

plt.axis('off')

plt.show()

**Task 3: To calculate disparity between two images (using stereoBM class).**

import numpy as np

import cv2

from matplotlib import pyplot as plt

imgL = cv2.imread('aloeL.jpg',0)

imgR = cv2.imread('aloeR.jpg',0)

stereo = cv2.StereoBM\_create(numDisparities=16, blockSize=15)

disparity = stereo.compute(imgL,imgR)

disparity1 = stereo.compute(imgR,imgL)

plt.subplot(2,2,1)

plt.title("aloe Left")

plt.imshow(imgL, cmap = 'gray')

plt.axis("off")

plt.subplot(2,2,2)

plt.title("aloe Right")

plt.imshow(imgR, cmap = 'gray')

plt.axis("off")

plt.subplot(2,2,(3,4))

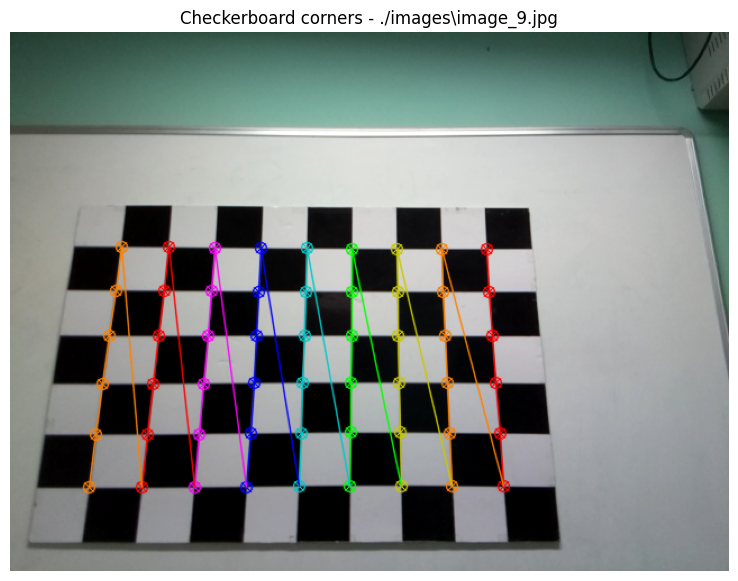
plt.title("Disparity Map")

plt.imshow(disparity,'gray')

plt.axis("off")

plt.show()

**Task 1: To implement camera calibration with Chessboard and Circular grid.**



Camera matrix:

[[503.68477278 0. 313.67563674]

[ 0. 503.37989194 243.25575476]

[ 0. 0. 1. ]]

Distortion coefficients:

[[ 2.08346324e-01 -4.68650266e-01 4.51079181e-04 -1.93373893e-03

2.37592401e-01]]

Rotation vectors:

(array([[-0.31060359],

[ 0.1515558 ],

[ 1.57720394]]), array([[0.3139181 ],

[0.58736866],

[1.39910762]]), array([[0.33307297],

[0.55841704],

[1.39532836]]), array([[0.78743976],

[0.53294469],

[1.52233962]]), array([[0.42519317],

[0.68127945],

[1.34442524]]), array([[-0.40676505],

[-0.72194813],

[ 1.36334531]]), array([[-0.41607396],

[-0.7647259 ],

[ 1.33850247]]), array([[-0.23343341],

[-0.04621596],

[ 1.522275 ]]), array([[ 0.11068043],

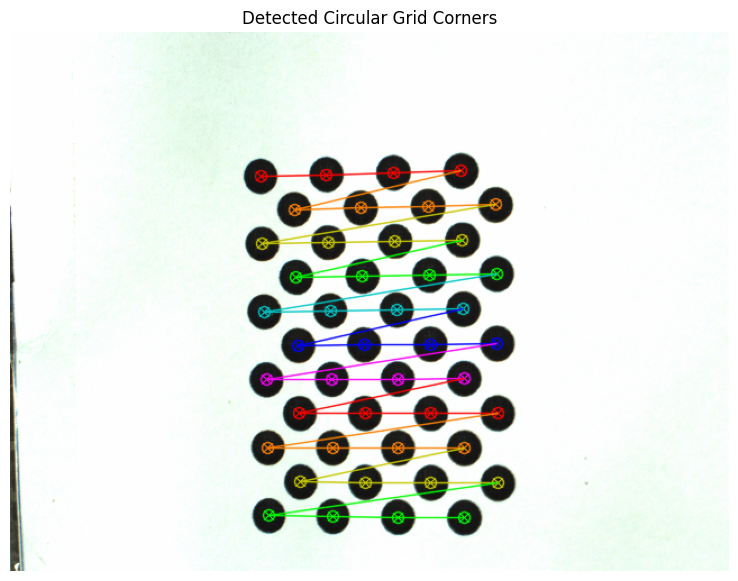
...

[-0.85351624],

[13.36840857]]), array([[ 2.75337631],

[-1.26097867],

[12.69840455]]))



Error in projection :

0.28397138993192417

Camera matrix :

[[ 2.98018946e+03 0.00000000e+00 -2.07790644e+02]

[ 0.00000000e+00 2.98680309e+03 5.80328416e+02]

[ 0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Distortion coefficients :

[[-1.38990879e+00 1.28121501e+01 -1.76642504e-02 4.92392900e-02

-6.65051660e+01]]

Rotation vector :

(array([[1.98945525],

[2.57035288],

[0.00544978]]), array([[-2.141599 ],

[-1.96709247],

[ 0.1071076 ]]), array([[-2.30654667],

[-1.91925376],

[-0.09466172]]), array([[-2.14337584],

[-1.80887689],

[-0.09635144]]), array([[-1.89996746],

[-1.89926953],

[ 0.33507546]]))

...

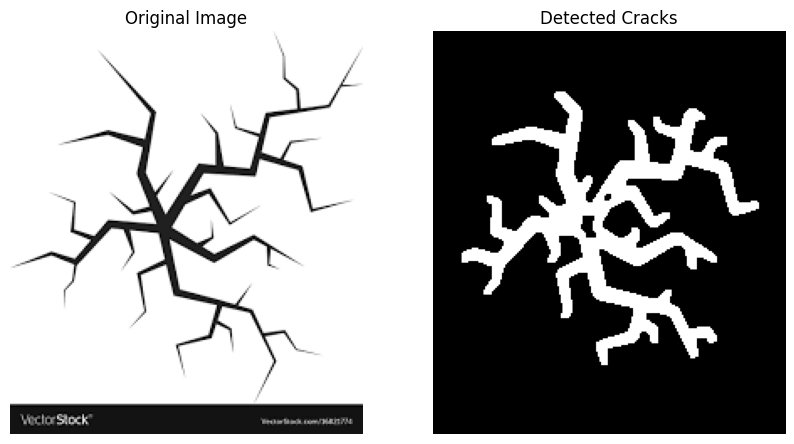
[-604.7774144 ],

[3469.45627266]]), array([[ 505.52036258],

[-533.92939056],

[3465.54276018]]))

**Task 2: To detect crack in the image using “shape from shading” technique.**



**Task 3: To calculate disparity between two images (using stereoBM class).**

