APPENDIX

Jika Anda punya lampiran dari project, silahkan dilampirkan di bagian ini. Yang wajib Anda lampirkan adalah kode program (coding) lengkap dan diberikan keterangan terlebih dahulu pada bagian atas dari coding tersebut, koding ditulis dengan format font yang berbeda. Contoh:

CONNECT TO DRIVE

```
220 !pwd
221
222 from google.colab import drive
223 drive.mount("/content/ProjectDataset/")
224
225 import os
226 os.chdir('/content/ProjectDataset/')
227 !pwd
```

COPY INPUT IMAGE TOP VIEW FROM DRIVE TO TMP FOLDER

```
228 import shutil
229 original=
    r'/content/ProjectDataset/MyDrive/ProjectDataset/InputImage/TestImage/Test2/top/lemon/lemon4_top1
    9.jpeg'
230 target = r'/tmp/inputtopview.jpg'
231
232 shutil.copyfile(original, target)
```

COPY INPUT IMAGE SIDE VIEW FROM DRIVE TO TMP FOLDER

```
233 import shutil
234
235 original=
    r'/content/ProjectDataset/MyDrive/ProjectDataset/InputImage/TestImage/Test2/side/lemon/lemon4_side
    19.jpeg'
236 target = r'/tmp/inputsideview.jpg'
237
238 shutil.copyfile(original, target)
```

REMOVE TOP VIEW INPUT IMAGE BACKGROUND TO TRANSPARENT

```
239 import numpy as np
240 import cv2 as cv
241 from matplotlib import pyplot as plt
242 import sys
243 from  PIL  import Image
244
245 img = cv.imread('/tmp/inputtopview.jpg', cv.IMREAD_UNCHANGED)
246 img = cv.rotate(img, cv.cv2.ROTATE_90_CLOCKWISE)
247 imagecopy = img.copy()
248
249 kernele = np.ones((8, 8), 'uint8')
```

```
kerneld = np.ones((4, 4), 'uint8')

filter = cv.cvtColor(img, cv.COLOR_BGR2HSV)
filter = cv.cvtColor(filter, cv.COLOR_HSV2RGB)
filter = cv.cvtColor(filter, cv.COLOR_RGB2GRAY)
filter = cv.GaussianBlur(filter, (17, 17), 17)
filter = cv.erode(filter, kernele, iterations=5)
filter = cv.Canny(filter, 23, 23)
filter = cv.dilate(filter, kerneld, iterations=5)

_, thresh = cv.threshold(filter, 0, 255, cv.THRESH_BINARY  + cv.THRESH_OTSU)
kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
mask = cv.morphologyEx(thresh, cv.MORPH_CLOSE, kernel, iterations=4)

data = mask.tolist()
sys.setrecursionlimit(10**8)
for i in  range(len(data)):
    for j in  range(len(data[i])):
        if data[i][j] !=  255:
            data[i][j] =  -1
        else:
            break
    for j in  range(len(data[i])-1, -1, -1):
        if data[i][j] !=  255:
            data[i][j] =  -1
        else:
            break
image = np.array(data)
image[image !=  -1] =  255
image[image ==  -1] =  0
mask = np.array(image, np.uint8)

main = cv.bitwise_and(imagecopy, imagecopy, mask=mask)
main[mask ==  0] =  255
cv.imwrite('/tmp/topviewextracted.jpg', main)
img = Image.open('/tmp/topviewextracted.jpg')
img.convert("RGBA")
datas = img.getdata()

datas2 = []
for item in datas:
    if item[0]==255 and item[1]==255 and item[2]==255:
        datas2.append((255, 255, 255, 0))
    else:
        datas2.append(item)

img.putdata(datas2)
img.save("/tmp/topviewextractedtr.png", "PNG")
plt.imshow(img)
```

REMOVE SIDE VIEW INPUT IMAGE BACKGROUND TO TRANSPARENT

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
import sys
from  PIL  import Image

```

```
305 img = cv.imread('/tmp/inputsideview.jpg', cv.IMREAD_UNCHANGED)
306 img = cv.rotate(img, cv.cv2.ROTATE_90_CLOCKWISE)
307 imagecopy = img.copy()
308
309 kernele = np.ones((7, 7), 'uint8')
310 kerneld = np.ones((3, 3), 'uint8')
311
312 filter = cv.cvtColor(img, cv.COLOR_BGR2HSV)
313 filter = cv.cvtColor(filter, cv.COLOR_HSV2RGB)
314 filter = cv.cvtColor(filter, cv.COLOR_RGB2GRAY)
315 filter = cv.GaussianBlur(filter, (15, 15), 15)
316 filter = cv.erode(filter, kernele, iterations=2)
317 filter = cv.Canny(filter, 25, 25)
318 filter = cv.dilate(filter, kerneld, iterations=2)
319
320 _, thresh = cv.threshold(filter, 0, 255, cv.THRESH_BINARY  + cv.THRESH_OTSU)
321 kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (5, 5))
322 mask = cv.morphologyEx(thresh, cv.MORPH_CLOSE, kernel, iterations=4)
323
324 data = mask.tolist()
325 sys.setrecursionlimit(10**8)
326 for i in  range(len(data)):
327     for j in  range(len(data[i])):
328         if data[i][j] !=  255:
329             data[i][j] =  -1
330         else:
331             break
332     for j in  range(len(data[i])-1, -1, -1):
333         if data[i][j] !=  255:
334             data[i][j] =  -1
335         else:
336             break
337 image = np.array(data)
338 image[image !=  -1] =  255
339 image[image ==  -1] =  0
340 mask = np.array(image, np.uint8)
341
342 main = cv.bitwise_and(imagecopy, imagecopy, mask=mask)
343 main[mask ==  0] =  255
344 cv.imwrite('/tmp/sideviewextracted.jpg', main)
345 img = Image.open('/tmp/sideviewextracted.jpg')
346 img.convert("RGBA")
347 datas = img.getdata()
348
349 datas2 = []
350 for item in datas:
351     if item[0]==255 and item[1]==255 and item[2]==255:
352         datas2.append((255, 255, 255, 0))
353     else:
354         datas2.append(item)
355
356 img.putdata(datas2)
357 img.save("/tmp/sideviewextractedtr.png", "PNG")
358 plt.imshow(img)
```

EXTRACT DATA TO TMP FOLDER

```
359 import zipfile
```

```
original_zip = '/content/ProjectDataset/MyDrive/ProjectDataset/Training10.zip'
zip_ref = zipfile.ZipFile(original_zip, 'r')
zip_ref.extractall('/tmp/Training')
zip_ref.close()
```

COPY TEST IMAGE SIDE VIEW FROM DRIVE TO TMP FOLDER

```
import shutil

local = r'/content/ProjectDataset/MyDrive/ProjectDataset/Input Image/Test Image/Test1/Test10/lemon.jpg'
os.mkdir('/tmp/Training/Input Image')
target = r'/tmp/Training/Input Image/inputimage.jpg'

shutil.copyfile(local, target)
```

STORE EXTRACTED RESULT IN TUPLE FORM

```
from pathlib import Path
import os
from collections import namedtuple
import pandas as pd

#all dataset
Files = namedtuple('File', 'name path')
dataset = []
p = Path('/tmp/Training/')
for item in p.glob('**/*'):
  name = item.name
  path = Path.resolve(item).parent
  dataset.append(Files(name, path))

dataset
```

REMOVE UNNECESSARY ROWS

```
import pandas as pd
all_datafiles = pd.DataFrame(dataset)
all_datafiles

all_files = all_datafiles.iloc[11:]
all_files
```

IMAGE HASH CALCULATION FUNCTION

```
!pip install imagehash
import imagehash
import numpy as np
from PIL import Image
from itertools import combinations
from sklearn.cluster import DBSCAN
from collections import defaultdict


def hashes_calculation(files, hashfunc=imagehash.whash):
    hashes, names = [], []
    for i, name in enumerate(files):
```

```
        try:
            img = Image.open(name)
            hash = hashfunc(img)
            hashes.append(hash)
            names.append(name)
        except:
            pass

    return hashes, names
```

MATRIX DISTANCE CALCULATION FUNCTION

```
def distances_calculation(hashes):
    matrix = np.zeros((len(hashes), len(hashes)))
    for i, j in combinations(range(len(hashes)), 2):
        dist = hashes[i] - hashes[j]
        matrix[i, j] = matrix[j,i] = dist
    return matrix
```

CALCULATE HASH FOR EACH IMAGE IN DATASET

```
from glob import glob
path = "/tmp/Training/**/*"
hashes, names = hashes_calculation(glob(path))
```

CALCULATE DISTANCE MATRIX FOR EACH HASHED IMAGE

```
matrix = distances_calculation(hashes)
matrix

import pandas as pd
df = pd.DataFrame(matrix)
df
```

DISTANCE MATRIX TO COORDINATE TRANSFORMATION

```
import numpy as np
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
dist2d = pca.fit_transform(matrix)
dist2d

import pandas as pd
df = pd.DataFrame(dist2d)
df.to_csv('/content/ProjectDataset/MyDrive/coordinates.csv',index=False)
```

CREATE DBSCAN CLASS

```
import numpy as np
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from itertools import cycle, islice
import matplotlib.pyplot as plt
import queue
import pandas as pd

```

```
445
446 class DBSCAN():
447     def __init__(self):
448         self.core = -1
449         self.border = -2
```

DBSCAN FIND NEIGHBOUR FUNCTION

```
450     def find_neighbour(self, data, point_id, eps):
451         points = []
452         for i in range(len(data)):
453             # Euclidian distance
454             if np.linalg.norm([a_i - b_i for a_i, b_i in zip(data[i], data[point_id])]) <= eps:
455                 points.append(i)
456         return points
```

DBSCAN FIT FUNCTION

```
457     def fit(self, data, Eps, MinPts):
458         point_label = [0] * len(data)
459         point_count = []
460
461         core = []
462         border = []
463
464         for i in range(len(data)):
465             point_count.append(self.find_neighbour(data, i, Eps))
466
467         for i in range(len(point_count)):
468             if (len(point_count[i]) >= MinPts):
469                 point_label[i] = self.core
470                 core.append(i)
471             else:
472                 border.append(i)
473
474         for i in border:
475             for j in point_count[i]:
476                 if j in core:
477                     point_label[i] = self.border
478                     break
479
480         cluster = 1
481
482         for i in range(len(point_label)):
483             q = queue.Queue()
484             if (point_label[i] == self.core):
485                 point_label[i] = cluster
486                 for x in point_count[i]:
487                     if(point_label[x] == self.core):
488                         q.put(x)
489                         point_label[x] = cluster
490                     elif(point_label[x] == self.border):
491                         point_label[x] = cluster
492                 while not q.empty():
493                     neighbors = point_count[q.get()]
494                     for y in neighbors:
495                         if (point_label[y] == self.core):
```

```
                    point_label[y] = cluster
                    q.put(y)
                if (point_label[y] == self.border):
                    point_label[y] = cluster
            cluster += 1  # Move on to the next cluster

    return point_label, cluster
```

DBSCAN VISUALIZATION FUNCTION

```
def visualize(self, data, cluster, clusters_count):
    N = len(data)

    colors = np.array(list(islice(cycle(['#FE4A49', '#2AB7CA']), 3)))

    for i in range(clusters_count):
        if (i == 0):
            color = '#000000'
        else:
            color = colors[i % len(colors)]

        x, y = [], []
        for j in range(N):
            if cluster[j] == i:
                x.append(data[j, 0])
                y.append(data[j, 1])
        plt.scatter(x, y, c=color, alpha=1, marker='.')
    plt.show()
```

DBSCAN CLUSTERING

```
df = pd.read_csv("/content/ProjectDataset/MyDrive/coordinates.csv")
dataset = df.astype(float).values.tolist()
X = StandardScaler().fit_transform(dataset)
DBSCAN = DBSCAN()
point_labels, clusters = DBSCAN.fit(X, 0.1, 2)
print(point_labels, clusters)
DBSCAN.visualize(X, point_labels, clusters)
print(clusters)
len(point_labels)
```

SELECT INPUT IMAGE CLUSTER ONLY

```
import pandas as pd
df = pd.DataFrame(point_labels, columns=['Point Labels'])
df

all_files.insert(2, "Point Labels", point_labels)
all_files

input_img = all_files[all_files['name'] == "inputimage.jpg"]

pntlbl = int(input_img['Point Labels'])
filter_data = all_files[all_files['Point Labels'] == pntlbl]
filter_data
```

NEW DATA FOR K-NN CONSIST OF IMAGES WITH THE SAME CLUSTER AS INPUT IMAGE

```
542 new_data = filter_data
543 new_data
544
545 new_data["filepath"] = new_data["path"] / new_data["name"]
546 new_data
547
548 new_data.reset_index(drop=True, inplace=True)
549 new_data
```

SWITCH THE FIRST ROW WITH INPUT IMAGE FOR K-NN REFERENCE POINT

```
550 switch = new_data.index[new_data['name'] == "inputimage.jpg"].tolist()
551 switch = switch[0]
552 switch
553
554 b = new_data.iloc[switch]
555 temp = new_data.iloc[0].copy()
556 new_data.iloc[0] = b
557 new_data.iloc[switch] = temp
558
559 new_data
```

IMAGE HASH CALCULATION

```
560 a = []
561 for i, row in new_data.iterrows():
562   knnpattern = str(row['filepath'])
563   hashes, fnames = hashes_calculation(glob(knnpattern))
564   b = list(hashes)
565   a = a + b
```

DISTANCE MATRIX CALCULATION

```
566 matrix2 = distances_calculation(a)
567
568 import pandas as pd
569 df = pd.DataFrame(matrix2)
570 df
```

DISTANCE MATRIX TO COORDINATE TRANSFORMATION

```
571 import numpy as np
572 from sklearn.decomposition import PCA
573 pca = PCA(n_components=2)
574 dist2d2 = pca.fit_transform(matrix2)
575
576 dist2d2
```

LABELLING IMAGES COORDINATES

```
577 import pandas as pd
578 df = pd.DataFrame(dist2d2)
```

```
df.columns =['X', 'Y']
df

result = pd.concat([df, new_data.reindex(df.index)], axis=1)
result
```

K-NN EUCLIDEAN DISTANCE FUNCTION

```
from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(vec1, vec2):
        distance = 0.0
        for i in range(len(vec1)-1):
                distance += (vec1[i] - vec2[i])**2
        return sqrt(distance)
```

K-NN GET NEIGHBOURS FUNCTION

```
def get_neighbors(train, test_row, num_neighbors):
        distances = list()
        for train_row in train:
                dist = euclidean_distance(test_row, train_row)
                distances.append((train_row, dist))
        distances.sort(key=lambda tup: tup[1])
        neighbors = list()
        for i in range(num_neighbors):
                neighbors.append(distances[i][0])
        return neighbors
```

K-NN CLASSIFICATION

```
n =  []
dataset = dist2d2
neighbors = list(get_neighbors(dataset, dataset[0], 2))
n = n + neighbors
print (n)
```

DESCRIBE THE RESULT OF THE NEAREST NEIGHBOUR

```
import pandas as pd
ans = pd.DataFrame(n)
ans.columns =['X', 'Y']
array = ans.to_numpy()
array

import numpy as np
res = array[1]
newres = np.array_split(res, 2)
newres
finalx = float(newres[0])
finalx

fin = result[np.isclose(result['X'],finalx, 0.0000000001)]
fin = fin.iloc[1: , :]
fin
```

EXTRACT THE OBJECT NAME

```
623 n = fin.iloc[:1]
624 n = n['path']
625 txt = n.to_string(index=False)
626 r = txt.split("/", 3)
627 fr = r[3]
628 fr
```

GET OBJECT DENSITY

```
629 import pandas as pd
630 dens = pd.read_csv("/content/ProjectDataset/MyDrive/ProjectDataset/density/Fruit_density.csv", sep=',')
631 dens
632
633 final_dense = dens[dens['Fruit'] == fr]
634 final_dense
635
636 density = float(final_dense['Density (g/cm3)'])
637 density
```

REPLACE TOP VIEW IMAGE TRANSPARENT BACKGROUND WITH BLACK

```
638 import sys
639 import numpy as np
640 import skimage.color
641 import skimage.filters
642 import skimage.io
643
644 image = skimage.io.imread(fname='/tmp/topviewextractedtr.png')
645 blur = skimage.color.rgb2gray(image)
646 mask = blur < 0.98
647 blackbg = np.zeros_like(image)
648 blackbg[mask] = image[mask]
649 skimage.io.imshow(blackbg)
```

OBJECT'S AREA CALCULATION

```
650 import cv2
651 import numpy as np
652 img = blackbg
653 height = img.shape[0]
654 width = img.shape[1]
655 gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
656 ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY)
657 plt.imshow(thresh)
658 count = cv2.countNonZero(thresh)
659 area = count*0.14*0.14/(width*height)
660 print(area)
```

REPLACE SIDE VIEW IMAGE TRANSPARENT BACKGROUND WITH BLACK

```
661 import sys
662 import numpy as np
663 import skimage.color
664 import skimage.filters
```

```
665 import skimage.io
666
667 image = skimage.io.imread(fname='/tmp/sideviewextractedtr.png')
668 skimage.io.imshow(image)
669 blur = skimage.color.rgb2gray(image)
670 mask = blur < 0.98
671 blackbg2 = np.zeros_like(image)
672 blackbg2[mask] = image[mask]
673
674 skimage.io.imshow(blackbg2)
```

OBJECT'S HEIGHT ESTIMATION

```
675 import cv2
676 import numpy as np
677 img = blackbg2
678 gray2 = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
679 ret, thresh2 = cv2.threshold(gray2,0,255,cv2.THRESH_BINARY)
680 plt.imshow(thresh2)
681
682 image2 = skimage.io.imread(fname="/tmp/sideviewextractedtr.png")
683 x,y,w,h = cv2.boundingRect(thresh2)
684 cv2.rectangle(image2, (x, y), (x + w, y + h), (36,255,12), 2)
685 cv2.putText(image2, "w={},h={}".format(w,h), (x,y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
    (36,255,12), 2)
686 plt.imshow(image2)
687 print(h)
```

OBJECT'S VOLUME CALCULATION

```
688 V = round(area*10000)*round((h/100)/2)
689 V
```

OBJECT'S MASS CALCULATION

```
690 M = density * V
691 M
692
693 round(M)
```

18.K1.0001.docx
Dec 6, 2021
15129 words / 75606 characters

Bonita Nugroho Widjanarko

18.K1.0001.docx

Sources Overview

9%

OVERALL SIMILARITY

1	arxiv.org INTERNET	2%
2	ntnuopen.ntnu.no INTERNET	1%
3	www.di.unipi.it INTERNET	1%
4	www.arpnjournals.com INTERNET	<1%
5	Min Jiang, Guodong Guo. "Body Weight Analysis From Human Body Images", IEEE Transactions on Information Forensics and Securit... CROSSREF	<1%
6	ebin.pub INTERNET	<1%
7	www.silsoeresearch.org.uk INTERNET	<1%
8	proceedings.mlr.press INTERNET	<1%
9	University of Computer Studies on 2016-08-12 SUBMITTED WORKS	<1%
10	K. Kollis, , C. S. Phang, T. M. Banhazi, and S. J. Searle. "Weight Estimation Using Image Analysis and Statistical Modelling: A Preliminar... CROSSREF	<1%
11	Nirav Alpesh Shah, Jaydeep Thik, Chintan Bhatt, Aboul-Ella Hassanien. "Chapter 4 A Deep Convolutional Encoder-Decoder Architectur... CROSSREF	<1%
12	ieeexplore.ieee.org INTERNET	<1%
13	Radhamadhab Dalai, Kishore Kumar Senapati. "A Heuristic Grid Area Based Segmentation Approach for Weight Estimation of an Obje... CROSSREF	<1%
14	Guihong Fu, Yun Yuna. "Phenotyping and phenomics in aquaculture breeding", Aquaculture and Fisheries, 2021 CROSSREF	<1%
15	KDU College Sdn Bhd on 2019-12-05 SUBMITTED WORKS	<1%

Excluded search repositories:

None

Excluded from document: