Daftar Pustaka

Armstrong, M. (2020). *The Vital Importance of Social Distancing*. Retrieved from https://www.statista.com/chart/21198/effect-of-social-distancing-signer-lab/.

Bubbling. (2020). *Object Detection YoloV4 (tensorflow2)*. Retrieved from https://blog.csdn.net/weixin_44791964/article/details/106533581

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *Yolov4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934.

Centers for Disease Control and Prevention (CDC). (2020). *Coronavirus Disease 2019 (COVID-19)*. Retrieved from https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/social-distancing.html.

Cohen, D. (2004), *Precalculus: A Problems-Oriented Approach* (6th ed.), Cengage Learning, p. 698, ISBN *978-0-534-40212-9*

Dhiaegana R. N. (2020). Penerapan Convolutional Neural Network Untuk Deteksi Pedestrian Pada Sistem Autonomous Vehicle (Skripsi). Sekolah Teknik Elektro dan Informatika. Institut Teknologi Bandung.

Fierro, A. N., Camarillo, D. R., & Miyatake, M. N. (2017). Mosquito Larva Classification Method Based on Convolutional Neural Networks. International Conference on Electronics, Communications and Computers (CONIELECOMP).

French, G., Fisher, M., Mackiewicz, M., & Needle, C. (2015). Convolutional Neural Networks for Counting Fish in Fisheries Surveillance Video. Workshop on Machine Vision of Animals and their Behaviour.

Geetha Kiran, A., & Murali, S. (2013). Automatic rectification of perspective distortion from a single image using plane homography. *Int. J. of Computational Sciences & Applications*, *3*(5), 47-58.

Ghiasi, G., Lin, T. Y., & Le, Q. V. (2018). Dropblock: A regularization method for convolutional networks. arXiv preprint arXiv:1810.12890.

Gibson, J. D., and Bovik A. (2000). Handbook of Image and Video Processing.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Huang, C., Wang, Y., Li, X., Ren, L., Zhao, J., Hu, Y., & Cao, B. (2020). Clinical Features of Patients Infected with 2019 Novel Coronavirus in Wuhan, China. The lancet, 395(10223), 497-506.

Hubel, D. H., & Wiesel, T. N. (1963). Shape and Arrangement of Columns in Cat's Striate Cortex. The Journal of physiology, 165(3), 559-568.

Jupiyandi, S., Saniputra, F., Pratama, Y., Dharmawan, M., & Cholissodin, I. (2019). "Pengembangan Deteksi Citra Mobil Untuk Mengetahui Jumlah Tempat Parkir Menggunakan CUDA dan Modified YOLO". Jurnal Teknologi Informasi dan Ilmu Komputer, 6(4), 413-419. doi: http://dx.doi.org/10.25126/jtiik.2019641275

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A Guide to Convolutional Neural Networks for Computer Vision. *Synthesis Lectures on Computer Vision*, *8*(1), 1-207. doi: 10.2200/s00822ed1v01y201712cov015.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., & Schindler, K. (2015). Motchallenge 2015: Towards a benchmark for multi-target tracking. arXiv *preprint arXiv:1504.01942.*

Liu, K. C., Xu, P., Lv, W. F., Qiu, X. H., Yao, J. L., Gu, J. F., & Wei, W. (2020). CT Manifestations of Coronavirus Disease-2019: a Retrospective Analysis of 73 Cases by Disease Severity. *European journal of radiology*, *126*, 108941.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Non-Linearities Improve Neural Network Acoustic Models. In Proc. icml (Vol. 30, No. 1, p. 3).

Manning, C. E. (2008). Digital Video Compression. Retrieved from http://www.newmediarepublic.com/dvideo/compression/adv01.html

Misra, D. (2019). Mish: A Self Regularized Non-Monotonic Neural Activation Dunction. arXiv preprint arXiv:1908.08681, 4.

Nalwan, A. (1997). *Pengolahan Gambar Secara Digital*. Jakarta: Elex Media Komputindo.

Nugraha, Y. A. (2014). Implementasi Sistem Otomatis pada Robot Kapal Berbasis Komputer Vision untuk Kontes Kapal Cepat Tak Berawak Nasional (KKCTBN).” Bandung: Universitas Komputer Indonesia.

Nurhikmat, T. (2018). Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek. Universitas Islam Indonesia

Padilla, R., Netto, S. L., & da Silva, E. A. (2020). A survey on performance metrics for object-detection algorithms. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 237-242). IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

Perkovic, L. (2012). Introduction to Computing Using Python: An Application Development Focus. J Wiley & Sons.

Permadi, Y., & Murinto, M. (2015). Aplikasi Pengolahan Citra Untuk Identifikasi Kematangan Mentimun Berdasarkan Tekstur Kulit Buah Menggunakan Metode Ekstraksi Ciri Statistik. Jurnal Informatika Ahmad Dahlan, 9(1), 103733. https://doi.org/10.26555/jifo.v9i1.a2044

Putri, R. K. S. C. (2018). Implementasi Deep Learning Menggunakan Metode Convolutional Neural Network Untuk Klasifikasi Gambar. (Skripsi). Universitas Islam Indonesia.

Read, P., & Meyer, M. P. (2000). Restoration of motion picture film. Elsevier.

Redmon, J. (2013). Darknet: Open Source Neural Networks in C. Retrieved from pjreddie.com/darknet/.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., & Sun, J. (2018). Crowdhuman: A Benchmark for Detecting Human in a Crowd. *arXiv preprint arXiv:1805.00123*.

Shin, K. G., & Ramanathan, P. (1994). Real-Time Computing: A New Discipline of Computer Science and Engineering. Proceedings of the IEEE. 82, pp. 6-24. IEEE. doi:10.1109/5.259423

Syahrudin, A. N., and T. Kurniawan. (2018). Input dan Output pada Bahasa Pemrograman Python. 7. Jurnal Dasar Pemograman Python STMIK.

Tan J. R. (2019). Breaking Down Mean Average Precision (mAP) - Towards Data Science. Retrieved from https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

Tuluran, J. (2020) Sistem Deteksi Pakan Udang Dengan Metode Deep Learning. (Skripsi). Departemen Teknik Elektro, Fakultas Teknik. Universitas Hasanuddin.

WHO. (2020). Coronavirus disease (COVID-19) Advice for The Public. Retrieved from https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public.

Wicaksono S., M. (2020). Pendeteksian dan Pelacakan Objek pada Lalu Lintas Padat dan Beragam untuk Mobil Otonom (Tesis). Sekolah Teknik Elektro dan Informatika. Institut Teknologi Bandung.

Worldometer. (2021) COVID-19 Coronavirus Pandemic. Worldometers.info, 2021, Retrieved from https://www.worldometers.info/coronavirus/.

Yao, Z., Cao, Y., Zheng, S., Huang, G., & Lin, S. (2020). Cross-Iteration Batch normalization. *arXiv preprint arXiv:2002.05712*.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 07, pp. 12993-13000).

LAMPIRAN

```
# import google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
# Nama folder untuk menyimpan log training, skor map, dan
juga bobot

SAVE_DIR = 'yolov4_crowdhuman'

# alamat folder pada google drive
DRIVE_DIR = '/content/drive/MyDrive'

# alamat menuh untuk menyimpan fodler dan file
DRIVE_SAVE_DIR = DRIVE_DIR + '/' + SAVE_DIR

# ukuran input dari model dan juga ukuran citra
INPUT_SHAPE = '416x416'
```

```
%cd /content
if not Path('yolov4_crowdhuman').is_dir():
    !git clone https://github.com/Marrz/yolov4_crowdhuman.git
```

```
#menjalankan script untuk mendownload dataset crowdhuman dan
melakukan konversi file
%cd /content/yolov4_crowdhuman/data
!./prepare_data.sh {INPUT_SHAPE}
```

```
#menyiapkan source code darknet
%cd /content/yolov4_crowdhuman
!rm -rf darknet
!git clone https://github.com/AlexeyAB/darknet.git
```

```
#menyiapkan darknet framework
%cd /content/yolov4_crowdhuman/darknet
!sed -i "1s/GPU=0/GPU=1/"               Makefile
!sed -i "2s/CUDNN=0/CUDNN=1/"           Makefile
!sed -i "3s/CUDNN_HALF=0/CUDNN_HALF=1/" Makefile
!sed -i "4s/OPENCV=0/OPENCV=1/"         Makefile
!sed -i "5s/AVX=0/AVX=1/"               Makefile
!sed -i "7s/LIBSO=0/LIBSO=1/"           Makefile
```

```
!sed -
i "20s/compute_30,code=sm_30 \\\/compute_37,code=sm_37 -
gencode arch=compute_60,code=[sm_60,compute_60] -
gencode arch=compute_61,code=[sm_61,compute_61] -
gencode arch=compute_75,code=[sm_75,compute_75]/" Makefile
!sed -i "21s/^/#/"                        Makefile
!sed -i "22s/^/#/"                        Makefile
!sed -i "23s/^/#/"                        Makefile
!sed -i "24s/^/#/"                        Makefile
# Let darknet test mAP more frequently during training
!sed -
i "300s/calc_map_for_each = 4/calc_map_for_each = 1/" src/det
ector.c

!make
```

```
#Membuat folder untuk menambung model dan training log
!mkdir -p "{DRIVE_SAVE_DIR}"
!rm -f "{DRIVE_SAVE_DIR}"/train.log
!touch "{DRIVE_SAVE_DIR}"/chart.png
!rm -f chart.png
!ln -sf "{DRIVE_SAVE_DIR}"/chart.png .
!mkdir -p "{DRIVE_SAVE_DIR}"/backup
!rm -rf backup
!ln -sf "{DRIVE_SAVE_DIR}"/backup .
```

```
#Menyiapkan file sebelum training
%cd /content/yolov4_crowdhuman
!./prepare_training.sh {INPUT_SHAPE}
```

```
#Memulai Training
%cd /content/yolov4_crowdhuman/darknet
!./darknet detector train data/crowdhuman-
{INPUT_SHAPE}.data cfg/yolov4-crowdhuman-
{INPUT_SHAPE}.cfg /content/drive/MyDrive/yolov4_crowdhuman/yo
lov4.conv.137 -gpus 0 -map -
dont_show 2>&1 | tee "{DRIVE_SAVE_DIR}"/train.log | grep -
E "hours left|mean_average"
```

```
#cek skor bobot
%cd /content/yolov4_crowdhuman/darknet
!./darknet detector map data/crowdhuman-
608x608.data cfg/yolov4-crowdhuman-
608x608.cfg /content/drive/MyDrive/yolov4_crowdhuman/backup/y
olov4-crowdhuman-608x608_6000.weights
```

```
#Tes Deteksi manusia
!./darknet detector demo data/crowdhuman-
608x608.data cfg/yolov4-crowdhuman-
608x608.cfg /content/drive/MyDrive/yolov4_crowdhuman/backup/y
olov4-crowdhuman-608x608_best.weights -
dont_show /content/drive/MyDrive/yolov4_crowdhuman/CCTV_Demo.
mp4 -i 0 -
out_filename /content/drive/MyDrive/yolov4_crowdhuman/results
.avi
```

```
#import cv2 lalu upgrade
import cv2
print(cv2.__version__)
!pip3 install --upgrade opencv-python==4.5.2.54
```

```
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt
import sys
import imutils
from imutils.video import FPS
```

```
#alamat model yolo dan juga konfigurasinya

configPath = '/content/drive/MyDrive/yolov4_crowdhuman/416x41
6/yolov4-crowdhuman-416x416.cfg'
weightsPath = '/content/drive/MyDrive/yolov4_crowdhuman/backu
p/yolov4-crowdhuman-416x416_6000.weights'
classesPath = '/content/drive/MyDrive/yolov4_crowdhuman/416x4
16/crowdhuman.names'
```

```
#mengekstrak frame pada video yang nanitnya digunakan untuk
melakukan kalibrasi
video_path = '/content/drive/MyDrive/yolov4_crowdhuman/Video/
Social Distancing Real Camera.mp4'
frame_path = '/content/drive/MyDrive/yolov4_crowdhuman/data/S
tudent_video_frame_{}.jpg'

frame_number = [100,200,300,400,500]

video = cv2.VideoCapture(video_path)

frame_count = 0
while True:
  ret,frame = video.read()

  if not ret:
    break

  frame_count+=1
  if frame_count in frame_number:
    print("Frame number : ",frame_count, "Saved")
    cv2.imwrite(frame_path.format(frame_count),frame)

video.release()
```

```
#memilih foto yang nantinya digunakan sebagaio acuan untuk
melakukan kalibrasi bird eye

original_image_BGR = cv2.imread('/content/drive/MyDrive/yolov
4_crowdhuman/Foto/2 Meter.jpg')
original_image_RGB = cv2.cvtColor(original_image_BGR, cv2.COL
OR_BGR2RGB)
main_header = cv2.imread('templates/main_header.jpg')

image_width = original_image_RGB.shape[1]
image_height = original_image_RGB.shape[0]

original_image_BGR_copy = original_image_BGR.copy()
original_image_RGB_copy = original_image_RGB.copy()

print('image Shape', original_image_RGB.shape)
```

```
#menentukan 4 titik sebagai acuan untuk melakukan
transformasi bird eye view
source_points = np.int_([[700, 777],
                         [1219, 776],
                         [1251, 892],
                         [ 667, 894]])
```

```
# memperlihatkan 4 titik yang telah dipilih

for point in source_points:
  cv2.circle(original_image_RGB_copy, point, 8, (255, 0, 0),
1)

points = source_points.reshape((-1,1,2)).astype(np.int32)
cv2.polylines(original_image_RGB_copy, [points], True, (0,255
,0), thickness=4)

plt.figure(figsize=(12, 12))
plt.imshow(original_image_RGB_copy)
plt.show()
```

```
#menentukan 4 titik sebagai acuan untuk melakukan
transformasi bird eye view
source_points = np.float32([[700, 777],
                         [1219, 776],
                         [1251, 892],
                         [ 667, 894]])
```

```
#membuat matriks homografi untuk transformasi citra menjadi
bird eye view

src=source_points
dst=np.float32([(0.22875,0.66111), (0.66125, 0.66111), (0.661
25,0.78148), (0.22875,0.78148)])

dst_size=(800,1080)
dst = dst * np.float32(dst_size)

H_matrix = cv2.getPerspectiveTransform(src, dst)
print("The perspective transform matrix:")
print(H_matrix)
```

```
#memperlihatkan hasil transformasi bird eye view

warped = cv2.warpPerspective(original_image_RGB_copy, H_matri
x, dst_size)

plt.figure(figsize=(12, 12))
plt.imshow(warped)
plt.show()
```

```
#mempersiapkan fungi yang diperlukan untuk melakukan deteksio
social distancing

import cv2
import numpy as np
from math import sqrt

img = []
ix,iy = 0,0
#membuat model darknet yang bisa berjalan pada cv2
def create_model(config, weights):

    model = cv2.dnn.readNetFromDarknet(config, weights)
    backend = cv2.dnn.DNN_BACKEND_OPENCV
    target = cv2.dnn.DNN_TARGET_CPU
    model.setPreferableBackend(backend)
    model.setPreferableTarget(target)
    return model

#output layer model
def get_output_layers(model):

    layer_names = model.getLayerNames()
    output_layers = [layer_names[i[0]-
1] for i in model.getUnconnectedOutLayers()]
    return output_layers
```

```
#melakukan normalisasi data pada citra
def blob_from_image(image, target_size):

    blob = cv2.dnn.blobFromImage(image, 1/255., target_size,
[0,0,0], 1, crop=False)
    return blob

#fungsi untuk melakukan prediski pada citra
def predict(blob, model, output_layers):

    model.setInput(blob)
    outputs = model.forward(output_layers)
    return outputs

def get_image_boxes(outputs, image_width, image_height, class
es, confidence_threshold=0.3, nms_threshold=0.3):
    class_ids = []
    confidences = []
    boxes = []
    results=[]

    for output in outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            class_name = classes[class_id]
            confidence = scores[class_id]
            if confidence > confidence_threshold and class_na
me== 'person':
                cx, cy, width, height = (detection[0:4] * np.
array([image_width, image_height, image_width, image_height])
).astype("int")
                x = int(cx - width / 2)
                y = int(cy - height / 2)
                boxes.append([x, y, int(width), int(height),c
x,cy])
                confidences.append(float(confidence))

    nms_indices = cv2.dnn.NMSBoxes(boxes, confidences, confid
ence_threshold, nms_threshold)
    if len(nms_indices)>0:
      return [boxes[ind] for ind in nms_indices.flatten()]
    else:
      return results
```

```
#fungsi untuk melakukan transformasi titik koordinat manusia
deteksi dengan transformasi perspetif.
def compute_point_perspective_transformation(matrix,boxes):
    list_downoids = [[box[4], box[5]+box[3]//2] for box in bo
xes]
    list_points_to_detect = np.float32(list_downoids).reshape
(-1, 1, 2)
    transformed_points = cv2.perspectiveTransform(list_points
_to_detect, matrix)
    transformed_points_list = list()
    for i in range(0,transformed_points.shape[0]):
        transformed_points_list.append([transformed_points[i]
[0][0],transformed_points[i][0][1]])
    return np.array(transformed_points_list).astype('int')

#fungsi untuk melakukan transformasi titik koordinat manusia
deteksi dengan transformasi perspetif.
def get_red_green_boxes(distance_allowed,birds_eye_points,box
es):
    red_boxes = []
    green_boxes = []

    new_boxes = [tuple(box) + tuple(result) for box, result i
n zip(boxes, birds_eye_points)]
    for i in range(0, len(new_boxes)-1):
            for j in range(i+1, len(new_boxes)):
                cxi,cyi = new_boxes[i][6:]
                cxj,cyj = new_boxes[j][6:]
                distance = eucledian_distance([cxi,cyi], [cxj
,cyj])
                if distance < distance_allowed:
                    red_boxes.append(new_boxes[i])
                    red_boxes.append(new_boxes[j])
                print(distance)
                print(distance*3.876)
    green_boxes = list(set(new_boxes) - set(red_boxes))
    red_boxes = list(set(red_boxes))

    return (green_boxes, red_boxes)

#kalkulasi eucledian distance
def eucledian_distance(point1, point2):
    x1,y1 = point1
    x2,y2 = point2
    return sqrt((x1-x2)**2 + (y1-y2)**2)
```

```
#tampilan bird eye view
def get_birds_eye_view_image(green_box, red_box,eye_view_heig
ht,eye_view_width):
    blank_image = cv2.imread('/content/drive/MyDrive/yolov4_c
rowdhuman/templates/black_background.png')

    cv2.putText(blank_image, str(len(red_box)), (120,100), cv
2.FONT_HERSHEY_SIMPLEX , 2, (0,0,255), 4, cv2.LINE_AA)
    cv2.putText(blank_image, str(len(green_box)), (520,100),
cv2.FONT_HERSHEY_SIMPLEX , 2, (0,255,0), 4, cv2.LINE_AA)

    for point in green_box:
        cv2.circle(blank_image,tuple([point[6],point[7]]),20,
(0,255,0),-1)
    for point in red_box:
        cv2.circle(blank_image,tuple([point[6],point[7]]),20,
(0,0,255),-1)
    blank_image = cv2.resize(blank_image,(eye_view_width,eye_
view_height))
    return blank_image

#menampilkan bounding box yang berwarna merah dan juga hijau
def get_red_green_box_image(new_box_image,green_box,red_box):
    for point in green_box:
        cv2.rectangle(new_box_image,(point[0],point[1]),(poin
t[0]+point[2],point[1]+point[3]),(0, 255, 0), 2)
    for point in red_box:
        cv2.rectangle(new_box_image,(point[0],point[1]),(poin
t[0]+point[2],point[1]+point[3]),(0, 0, 255), 2)
    return new_box_image
```

```
#menyiapkan bobot model YOLO dan menyiapkan parameter yang
diperlukan
confidence_threshold = 0.5
nms_threshold = 0.4

min_distance = 228
width = 416
height = 416

config = configPath
weights = weightsPath
classes = classesPath

with open(classes, 'rt') as f:
    coco_classes = f.read().strip('\n').split('\n')

model = create_model(config, weights)
output_layers = get_output_layers(model)
```

```
#Deteksi social distancing pada video digital
%%time
print("[INFO] accessing video stream...")
video = cv2.VideoCapture('/content/drive/MyDrive/yolov4_crowd
human/Video/Social Distancing Real Camera.mp4')
fps = FPS().start()
writer = None
frame_number = 0
print("[INFO] Detecting Social Distancing...")
print('%-20s%-26s%-26s%-
26s' % ('Processing Frame','|Total Detected Person','|Red Mar
kerd Person','|Green Marked Person'))

while True:

  ret,frame = video.read()

  if not ret:
    break

  image_height, image_width = frame.shape[:2]

  image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
  blob = blob_from_image(image, (width, height))
```

```
  outputs = predict(blob, model, output_layers)

  boxes = get_image_boxes(outputs, image_width, image_height,
 coco_classes)
  birds_eye_points = compute_point_perspective_transformation
(H_matrix, boxes)
  green_box, red_box = get_red_green_boxes(min_distance, bird
s_eye_points,boxes)
  birds_eye_view_image = get_birds_eye_view_image(green_box,
red_box, eye_view_height=image_height,eye_view_width=image_wi
dth//2)
  box_red_green_image = get_red_green_box_image(frame.copy(),
 green_box,red_box)

  combined_image = np.concatenate((birds_eye_view_image,box_r
ed_green_image), axis=1)

  frame_number += 1
  sys.stdout.write('%-20i|%-25i|%-25i|%-
25i\n' % (frame_number,len(boxes),len(red_box),len(green_box)
))

  #if frame_number >=120:
    #break

  if writer is None:
    fourcc = cv2.VideoWriter_fourcc(*"MJPG")
    writer = cv2.VideoWriter('/content/Social Distance With C
amera.avi', fourcc, 60, (combined_image.shape[1], combined_im
age.shape[0]), True)

  writer.write(combined_image)

  del image,outputs,combined_image,birds_eye_view_image
  fps.update()



fps.stop()
print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
print("Done")
print(' ')
writer.release()
video.release()
```