DAFTAR PUSTAKA

Annur, C. M. (2023, July 18). Pencurian, kejahatan paling banyak di Indonesia sampai April 2023. Databoks. https://databoks.katadata.co.id/datapublish/2023/07/18/ pencurian-kejahatan-paling-banyak-di-indonesia-sampai-april- 2023

Al Najjar, Y. (2024). Comparative analysis of image quality assessment metrics: MSE, PSNR, SSIM and FSIM. International Journal of Science and Research (IJSR), 13(3).

Aji, B. B. P., Istikmal, & Irawan, A. I. (2023). Implementasi ESP32-CAM dan Aplikasi Blynk pada Smart Door Bell Sistem. Jurnal Teknik Elektro Telkom University, 12(1), 1-10.

Dorothy, R., Joany, R. M., Rathish, R. J., Prabha, S. S., & Rajendran, S. (2015). Image enhancement by *histogram equalization*. International Journal of Nano Corrosion Science and Engineering, 2(4), 21-30. ISSN Online: 2395-7018.

Furqan, M., Armansyah, & Muliani, N. H. (2022). Application of *contrast*-limited adaptive *histogram equalization* (CLAHE) and *Gaussian filter* methods for improvement of image quality on closed-circuit television (CCTV). Jurnal INFOKUM, 10(4), 119.

Geraldy, C., & Lubis, C. (2024). Pendeteksian dan Pengenalan Jenis Mobil Menggunakan Algoritma *You Only Look Once* dan Convolutional Neural Network. Jurnal Ilmu Komputer dan Sistem Informasi (JIKSI), 197.

Junaidi, A. (2015). *Internet of Things*: Sejarah, Teknologi, dan Penerapannya: Review. Jurnal Ilmiah Teknologi Informasi Terapan, 1(3), 62–66.

Kamal, Firdayanti, Ulfa Mahanin Tyas, Andi Apri Buckhari, dan Pattasang. “Implementasi Aplikasi Arduino IDE pada Mata Kuliah Sistem Digital.” Setyaki Jurnal Studi Keagamaan Islam, Volume 1, Nomor 1, April 2023, E-ISSN: 2987-07471

Kamil, F. (2023). Pengolahan Citra Digital Menggunakan Metode YOLO untuk Mendeteksi Kualitas Dari Biji Kopi Berbasis Android. Jurnal AI dan SPK: Jurnal Artificial Intelligent dan Sistem Penunjang Keputusan, 1(1), 120-125. DOI: 10.24843/jaispk.v1i1.2421

Kapoor, K., & Arora, S. (2015). Colour image enhancement based on *histogram equalization*. Electrical & Computer Engineering: An International Journal (ECIJ), 4(3), 73. https://doi.org/10.14810/ecij.2015.4306

Kusuma, I. W. A. W., & Kusumadewi, A. (2020). Penerapan metode *contrast* stretching, *histogram equalization*, dan adaptive *histogram equalization* untuk meningkatkan kualitas citra medis MRI. Jurnal SIMETRIS, 11(1), 1-10. P-ISSN: 2252-4983, E-ISSN: 2549-3108.

Noviyanti, D. S. (2020). Penerapan Metode *Contrast* Limited Adaptive *Histogram equalization* (CLAHE) dan *Gaussian filter* dalam Perbaikan Kualitas Citra Fundus

Retina. Tugas Akhir, Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Sriwijaya.

Nabuasa, Y. N. (2019). Perbandingan Metode *Histogram equalization* dan Specification pada Citra Abu-Abu. J-ICON, 7(1), 87–95. ISSN: 2337-7631 (Printed), ISSN: 2654-4091 (Online).

Prastya, D. Z. E., Pamungkas, D. P., & Niswatin, R. K. (2022, July 23). Implementasi metode *Gaussian filter* dan *Median filter* untuk penghalusan gambar. Seminar Nasional Inovasi Teknologi, Universitas Nusantara PGRI Kediri, 178-184. https://doi.org/e-ISSN:2549-7952

Peng, J., Shi, C., Laugeman, E., Hu, W., Zhang, Z., Mutic, S., & Cai, B. (2020). Implementation of the Structural SIMilarity (SSIM) index as a quantitative evaluation tool for dose distribution error detection. Medical Physics. https://doi.org/10.1002/mp.14010

Romzi, M., & Kurniawan, B. (2020). Implementasi Pemrograman Python menggunakan *Visual Studio Code*. Jurnal Informatika dan Komputer (JIK), 11(2), 1-9.

Sary, I. P., Armin, E. U., & Andromeda, S. (2023). *Performance* comparison of YOLOv5 and YOLOv8 architectures in human detection using aerial images. Jurnal Elektronika, Universitas Singaperbangsa Karawang. ISSN 2355-3286.

Sara, U., Akter, M., & Uddin, M. S. (2019). Image quality assessment through FSIM, SSIM, MSE and PSNR—A comparative study. Journal of Computer and Communications, 7(3), 1-10.

Siswanto, S., Nurhadiyan, T., & Junaedi, M. (2020). Prototype Smart Home dengan Konsep IoT (*Internet of Things*) Berbasis NodeMCU dan Telegram. Jurnal Sistem Informasi dan Informatika (Simika), 3(1), 85-93. https://doi.org/10.47080/simika.v3i1.850

Saputra, F. A., & Chandra, J. C. (2022). Prototipe Sistem Keamanan Ruang *Server* Otomatis Menggunakan ESP32CAM dan Algoritma *You Only Look Once* (YOLO). Jurnal TICOM: Technology of Information and Communication, 11(1), 62-67. E-ISSN: 2962-7982.

Venna, F. C., & Tjahjanto. (2022). Perbandingan IoT pada Sensor Kinect, Sensor PIR, dan RFID dalam Sistem Keamanan Rumah. Just IT: Jurnal Sistem Informasi, Teknologi Informasi dan Komputer, 13(1), 22-29. P-ISSN: 2089-0256, e-ISSN: 2598-3016.

Wilianto, & Kurniawan, A. (2018). Sejarah, Cara Kerja, dan Manfaat *Internet of Things*. Jurnal Matrix, 8(2), Juli.

Wang, M., Zheng, S., Li, X., & Qin, X. (2014). A new image denoising method based on *Gaussian filter*. College of Computer and Information Technology, China Three Gorges University, Yichang, Hubei, China.

Yoga, I. P. S., Sukadarmika, G., Hartati, R. S., & Divayana, Y. (2023). Pendeteksi Jumlah Orang pada Sistem Bangunan Pintar Menggunakan Algoritma *You Only Look Once*. Majalah Ilmiah Teknologi Elektro, 22(1), 11-18. DOI: 10.24843/MITE.2023.v22i01.P02

Zuanita Syifaul Jannah, & Felix Andreas Sutanto. (2022). Implementasi Algoritma YOLO (*You Only Look Once*) Untuk Deteksi Rias Adat Nusantara. Jurnal Ilmiah Universitas Batanghari Jambi, 22(3), 1490-1495. DOI: 10.33087/jiubj.v22i3.2421

Zoumana Keita, "YOLO Object Detection Explained" Datacamp (2020), diakses pada 5 Juni 2024, https://www.Datacamp.com/blog/yolo-object-detection-explained

LAMPIRAN

Lampiran 1 Skrip program pengukuran metriks performance

Skrip program ini memakai Bahasa pemrograman python dan dijalankan pada visual studio code.

Skrip pengukuran YOLOv4 dengan *filter Pre-processing*

```
# Load YOLOv4 model using OpenCV

import os

import cv2

import numpy as np

from pycocotools.coco import COCO


# Path setup

coco_annotation_file = os.path.join('path', 'to', 'annotations', 'instances_val2017.json')

image_folder = os.path.join('path', 'to', 'images', 'val2017')

model_folder = os.path.dirname(os.path.abspath(__file__))


net = cv2.dnn.readNetFromDarknet(os.path.join(model_folder, 'yolov4.cfg'), os.path.join(model_folder, 'yolov4.weights'))

layer_names = net.getLayerNames()

output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]


# Preprocess image with optimized filters for YOLOv4

def preprocess_image_with_optimized_filters(image_path, target_size):

    if not os.path.exists(image_path):

        raise FileNotFoundError(f"File {image_path} tidak ditemukan.")


    image = cv2.imread(image_path)

    if image is None:

        raise IOError(f"Gagal membaca file gambar {image_path}. Periksa jalur file dan integritasnya.")
```

```
    # Calculate brightness
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    brightness = hsv[..., 2].mean()

    # Set filter parameters based on brightness
    if brightness < 50:
        blur_strength = int(np.interp(brightness, [0, 50], [15, 5]))
        CLAHE_limit = np.interp(brightness, [0, 50], [4.0, 2.0])
        tile_grid_size = int(np.interp(brightness, [0, 50], [16, 8]))

        # Ensure blur_strength is odd
        blur_strength = max(1, blur_strength | 1)

        image = cv2.GaussianBlur(image, (blur_strength, blur_strength), 0)
        image = cv2.medianBlur(image, blur_strength)

        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        equalized = cv2.equalizeHist(gray)

        CLAHE = cv2.createCLAHE(clipLimit=CLAHE_limit,
tileGridSize=(tile_grid_size, tile_grid_size))
        CLAHE_applied = CLAHE.apply(equalized)

        # Convert to grayscale if very dark
        if brightness < 20:
            image = cv2.cvtColor(CLAHE_applied, cv2.COLOR_GRAY2BGR)
        else:
            lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
            l, a, b = cv2.split(lab)
```

```
            l = CLAHE.apply(l)
            lab = cv2.merge((l, a, b))
            image = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    else:
        # Apply minimal filtering to maintain colors
        image = cv2.GaussianBlur(image, (3, 3), 0)
        image = cv2.medianBlur(image, 3)

    blob = cv2.dnn.blobFromImage(image, 1/255.0, target_size, swapRB=True, crop=False)
    return blob, image

# Perform detection using YOLOv4
def detect_objects(image, net, output_layers):
    net.setInput(image)
    outputs = net.forward(output_layers)
    return outputs

# Postprocess detections
def postprocess_detections(outputs, width, height, conf_threshold=0.5, nms_threshold=0.4):
    boxes = []
    confidences = []
    class_ids = []
    for output in outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > conf_threshold:
```

```
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, x + w, y + h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
    indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
    return [(boxes[i], class_ids[i]) for i in indices]


# Evaluate detections manually
def evaluate_detections(detections, ground_truths, iou_threshold=0.3):
    tp = 0
    fp = 0
    fn = len(ground_truths)

    for det in detections:
        detected = False
        for gt in ground_truths:
            iou = calculate_iou(gt, det[0])
            if iou > iou_threshold:
                detected = True
                tp += 1
                fn -= 1
                break
        if not detected:
            fp += 1
```

```
    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else
0

    return precision, recall, f1_score

# Calculate IoU (Intersection over Union)
def calculate_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])

    interArea = max(0, xB - xA) * max(0, yB - yA)
    boxAArea = (boxA[2] - boxA[0]) * (boxA[3] - boxA[1])
    boxBArea = (boxB[2] - boxB[0]) * (boxB[3] - boxB[1])

    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou
```

Skrip pengukuran YOLOv4 tanpa *filter Pre-processing*

```
# Load YOLOv4 model using OpenCV
import os
import cv2
import numpy as np
import pandas as pd
import tkinter as tk
from tkinter import ttk, messagebox
from pycocotools.coco import COCO
```

```
# Path setup
coco_annotation_file = os.path.join('path', 'to', 'annotations', 'instances_val2017.json')
image_folder = os.path.join('path', 'to', 'images', 'val2017')
model_folder = os.path.dirname(os.path.abspath(__file__))

# Load YOLOv4 model using OpenCV
net = cv2.dnn.readNetFromDarknet(os.path.join(model_folder, 'yolov4.cfg'), os.path.join(model_folder, 'yolov4.weights'))
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Load COCO annotations
coco = COCO(coco_annotation_file)

# Load class names
def load_class_names(file_path):
    with open(file_path, 'r') as f:
        class_names = f.read().strip().split('\n')
    return class_names

# Preprocess image for YOLOv4 (without additional filters)
def preprocess_image(image_path, target_size):
    image = cv2.imread(image_path)
    blob = cv2.dnn.blobFromImage(image, 1/255.0, target_size, swapRB=True, crop=False)
    return blob, image

# Perform detection using YOLOv4
def detect_objects(image, net, output_layers):
```

```
    net.setInput(image)
    outputs = net.forward(output_layers)
    return outputs


# Postprocess detections
def postprocess_detections(outputs, width, height, conf_threshold=0.5, nms_threshold=0.4):
    boxes = []
    confidences = []
    class_ids = []
    for output in outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > conf_threshold:
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)
                boxes.append([x, y, x + w, y + h])
                confidences.append(float(confidence))
                class_ids.append(class_id)
    indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
    return [(boxes[i], class_ids[i]) for i in indices]


# Evaluate detections manually
```

```
def evaluate_detections(detections, ground_truths, iou_threshold=0.3):
    tp = 0
    fp = 0
    fn = len(ground_truths)

    for det in detections:
        detected = False
        for gt in ground_truths:
            iou = calculate_iou(gt, det[0])
            if iou > iou_threshold:
                detected = True
                tp += 1
                fn -= 1
                break
        if not detected:
            fp += 1

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else
0

    return precision, recall, f1_score

# Calculate IoU (Intersection over Union)
def calculate_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
```

```
    interArea = max(0, xB - xA) * max(0, yB - yA)
    boxAArea = (boxA[2] - boxA[0]) * (boxA[3] - boxA[1])
    boxBArea = (boxB[2] - boxB[0]) * (boxB[3] - boxB[1])

    iou = interArea / float(boxAArea + boxBArea - interArea)
    return iou
```

Lampiran 2 Skrip program parameter kualitas gambar

```
import os
import cv2
import numpy as np
import pandas as pd
from skimage.metrics import structural_similarity as ssim
from math import log10, sqrt
import warnings

# Image processing parameters
blur_strength_low = 3
blur_strength_high = 15
CLAHE_limit_low = 2.0
CLAHE_limit_high = 4.0
tile_grid_size_low = 8
tile_grid_size_high = 16

# Folder containing images
image_folder = os.path.join('path', 'to', 'images', 'val2017')

# Function to apply pre-processing filters
def apply_filters(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

```
    brightness = hsv[..., 2].mean()


    if brightness < 50:
        blur_strength = int(np.interp(brightness, [0, 50], [blur_strength_high,
blur_strength_low]))
        CLAHE_limit = np.interp(brightness, [0, 50], [CLAHE_limit_high,
CLAHE_limit_low])
        tile_grid_size = int(np.interp(brightness, [0, 50], [tile_grid_size_high,
tile_grid_size_low]))
        blur_strength = max(1, blur_strength | 1)


        image = cv2.GaussianBlur(image, (blur_strength, blur_strength), 0)
        image = cv2.medianBlur(image, blur_strength)


        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        equalized = cv2.equalizeHist(gray)


        CLAHE = cv2.createCLAHE(clipLimit=CLAHE_limit,
tileGridSize=(tile_grid_size, tile_grid_size))
        CLAHE_applied = CLAHE.apply(equalized)


        if brightness < 20:
            image = cv2.cvtColor(CLAHE_applied, cv2.COLOR_GRAY2BGR)
        else:
            lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
            l, a, b = cv2.split(lab)
            l = CLAHE.apply(l)
            lab = cv2.merge((l, a, b))
            image = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    else:
        image = cv2.GaussianBlur(image, (blur_strength_low, blur_strength_low), 0)
        image = cv2.medianBlur(image, blur_strength_low)
```

```
    return image

# Function to calculate SNR
def calculate_snr(image, noise):
    signal_power = np.mean(image ** 2)
    noise_power = np.mean(noise ** 2)
    if noise_power == 0:
        return float('inf')
    snr_value = 10 * log10(signal_power / noise_power)
    return snr_value

# Function to calculate PSNR
def calculate_psnr(original, processed):
    mse_value = np.mean((original - processed) ** 2)
    if mse_value == 0:
        return float('inf')
    max_pixel = 255.0
    psnr_value = 20 * log10(max_pixel / sqrt(mse_value))
    return psnr_value

# Function to calculate MSE
def calculate_mse(original, processed):
    mse_value = np.mean((original - processed) ** 2)
    return mse_value

# Function to calculate SSIM
def calculate_ssim(original, processed):
    min_side = min(original.shape[0], original.shape[1])
    win_size = min(7, min_side)
    ssim_value = ssim(original, processed, channel_axis=-1, win_size=win_size)
```

```
    return ssim_value
```

Lampiran 3 Skrip program untuk menjalankan pengujian secara *realtime*

```
import cv2
import numpy as np
import URLlib.request
import requests
import time
from telegram import Bot

# YOLO Model Configuration
whT = 320
confThreshold = 0.15  # Lowering confidence threshold
nmsThreshold = 0.05  # Lowering NMS threshold
classesfile = 'path/to/coco.names'
classNames = []

with open(classesfile, 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')

modelConfig = 'path/to/yolov4.cfg'
modelWeights = 'path/to/yolov4.weights'
net = cv2.dnn.readNetFromDarknet(modelConfig, modelWeights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

# Telegram Bot Configuration
telegram_bot_token = 'YOUR_TELEGRAM_BOT_TOKEN'
telegram_chat_id = 'YOUR_TELEGRAM_CHAT_ID'
bot = Bot(token=telegram_bot_token)

# Camera URL Configuration
```

```
camera_URL = 'http://path/to/camera_feed.jpg'

# Set up a session with retry strategy
session = requests.Session()

# Function to calculate brightness
def calculate_brightness(im):
    hsv = cv2.cvtColor(im, cv2.COLOR_BGR2HSV)
    brightness = hsv[..., 2].mean()
    return brightness

# Function to apply dynamic filters
def apply_dynamic_filters(im):
    brightness = calculate_brightness(im)
    if brightness < 20:
        blur_strength = int(np.interp(brightness, [0, 20], [15, 5]))
        CLAHE_limit = np.interp(brightness, [0, 20], [4.0, 2.0])
        tile_grid_size = int(np.interp(brightness, [0, 20], [16, 8]))

        if blur_strength % 2 == 0:
            blur_strength += 1
        if blur_strength < 1:
            blur_strength = 1

        im = cv2.GaussianBlur(im, (blur_strength, blur_strength), 0)
        im = cv2.medianBlur(im, blur_strength)

        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
        equalized = cv2.equalizeHist(gray)
```

```
        CLAHE = cv2.createCLAHE(clipLimit=CLAHE_limit,
tileGridSize=(tile_grid_size, tile_grid_size))
        CLAHE_applied = CLAHE.apply(equalized)


        im = cv2.cvtColor(CLAHE_applied, cv2.COLOR_GRAY2BGR)
    else:
        im = cv2.GaussianBlur(im, (3, 3), 0)
        im = cv2.medianBlur(im, 3)


    return im, brightness


# Function to find objects using YOLOv4
def findObject(outputs, im, with_filter):
    hT, wT, cT = im.shape
    bbox = []
    classIds = []
    confs = []
    found_person = False
    for output in outputs:
        for det in output:
            scores = det[5:]
            classId = np.argmax(scores)
            confidence = scores[classId]
            if confidence > confThreshold:
                w, h = int(det[2] * wT), int(det[3] * hT)
                x, y = int((det[0] * wT) - w / 2), int((det[1] * hT) - h / 2)
                bbox.append([x, y, w, h])
                classIds.append(classId)
                confs.append(float(confidence))


    indices = cv2.dnn.NMSBoxes(bbox, confs, confThreshold, nmsThreshold)
```

```
    if len(indices) > 0:
        for i in indices.flatten():
            box = bbox[i]
            x, y, w, h = box[0], box[1], box[2], box[3]
            if classNames[classIds[i]] == 'person':
                found_person = True
            cv2.rectangle(im, (x, y), (x + w, y + h), (255, 0, 255), 1)
            text = f'{classNames[classIds[i]].upper()} {int(confs[i] * 100)}%'
            cv2.putText(im, text, (x, y - 10 if y - 10 > 10 else y + h + 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 255), 1)
        return found_person
    return False


# Main Loop to Capture Image and Perform Detection
while True:
    try:
        img_resp = URLlib.request.URLopen(camera_URL)
        imgnp = np.array(bytearray(img_resp.read()), dtype=np.uint8)
        im = cv2.imdecode(imgnp, -1)


        # Resize image for faster processing
        im_small = cv2.resize(im, (whT, whT))


        # Apply dynamic filters
        im_with_filter, brightness_with_filter = apply_dynamic_filters(im_small.copy())


        blob_with_filter = cv2.dnn.blobFromImage(im_with_filter, 1/255, (whT, whT),
[0, 0, 0], 1, crop=False)
        net.setInput(blob_with_filter)
        outputs_with_filter = net.forward([net.getLayerNames()[i - 1] for i in
net.getUnconnectedOutLayers()])
```

```
        detected_with_filter = findObject(outputs_with_filter, im_with_filter,
with_filter=True)
        if detected_with_filter:
            print("Person detected with filter")
            # Additional actions can be added here (e.g., send notification)
        # Show the processed image
        cv2.imshow('Image with Filter', im_with_filter
        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):
            break
    except Exception as e:
        print(f"Error: {e}")
        break
cv2.destroyAllWindows()
```

Lampiran 4 Dokumentasi pengujian secara *realtime*

Dokumentasi Pengkalibrasian Kamera *ESP32*-CAM Dalam Ruangan

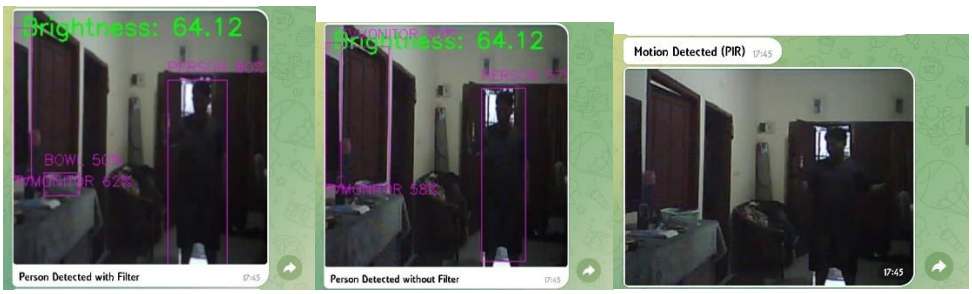
Dokumentasi Pengkalibrasian Kamera *ESP32*-CAM Luar Ruangan

Pemasangan Alat Untuk Pengujian Dalam Ruangan

Pemasangan Alat Untuk Pengujian Luar Ruangan

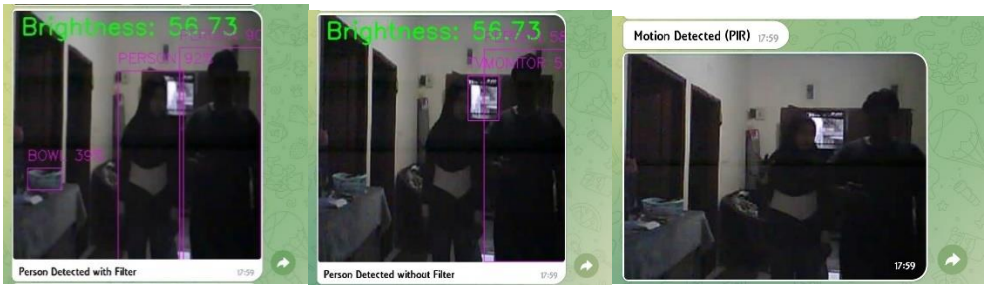
Pengujian Pemakaian Secara Real Time Dalam Ruangan

(Dok. Percobaan Jarak 2 Meter Dalam Ruangan Objek 1 Orang)

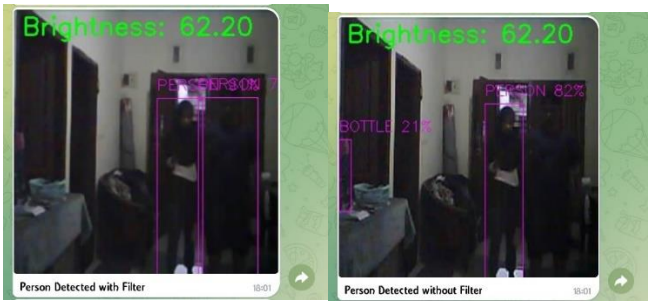


(Dok. Percobaan Jarak 3.6 Meter Dalam Ruangan Objek 1 Orang)

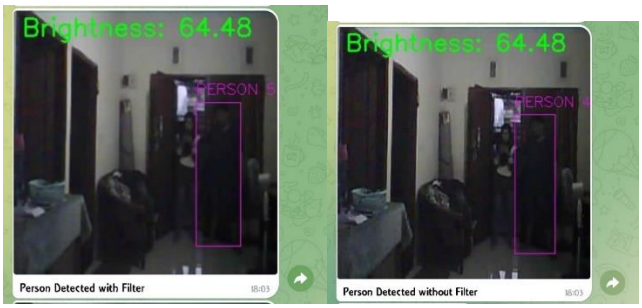
(Dok. Percobaan Jarak 5.1 Meter Dalam Ruangan Objek 1 Orang)



(Dok. Percobaan Jarak 2 Meter Dalam Ruangan Objek 2 Orang)

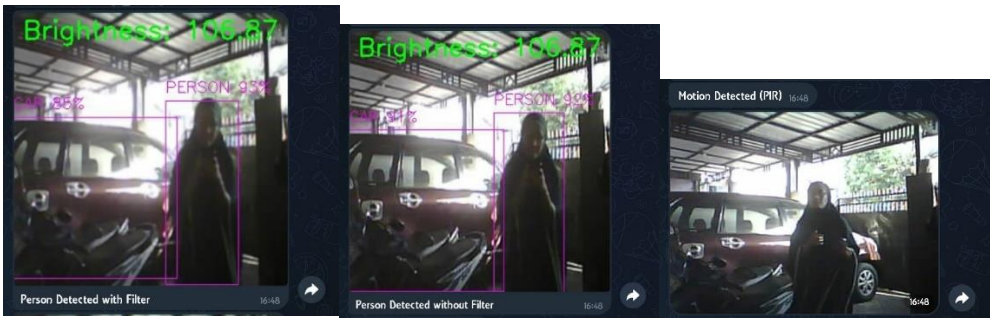


(Dok. Percobaan Jarak 3.6 Meter Dalam Ruangan Objek 2 Orang)

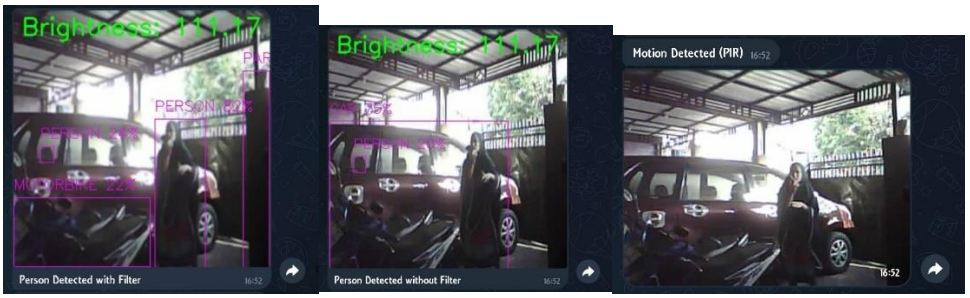


(Dok. Percobaan Jarak 5.1 Meter Dalam Ruangan Objek 2 Orang)

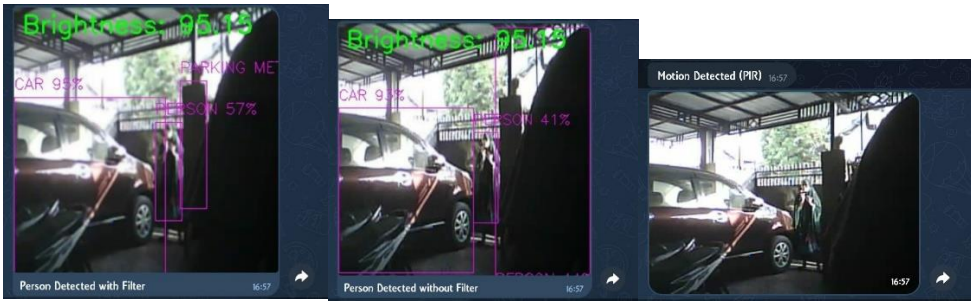
Pengujian Pemakaian Secara Real Time Diluar Ruangan (Kondisi Siang-Sore)



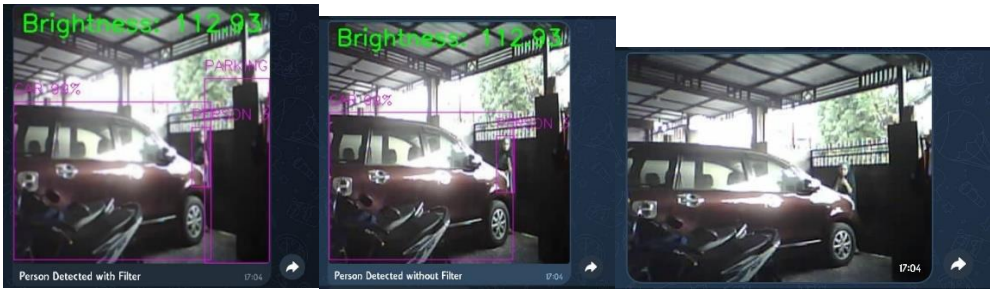
(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 1 Orang)



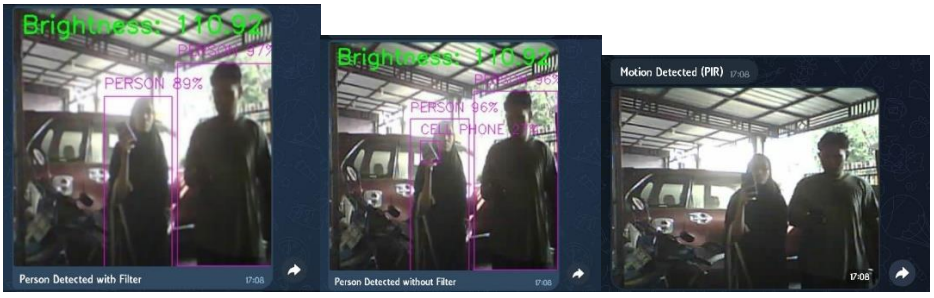
(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 1 Orang)



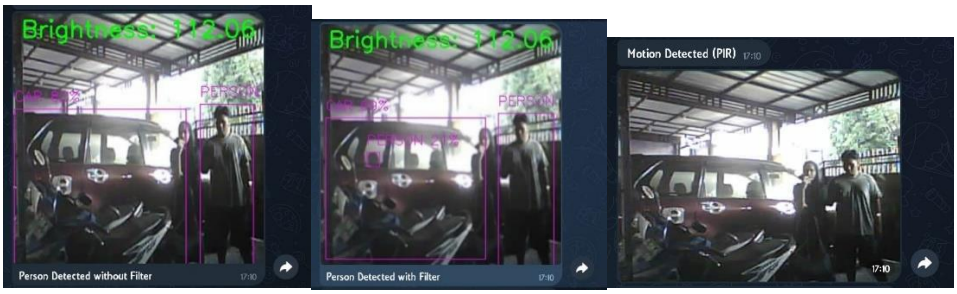
(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 1 Orang)



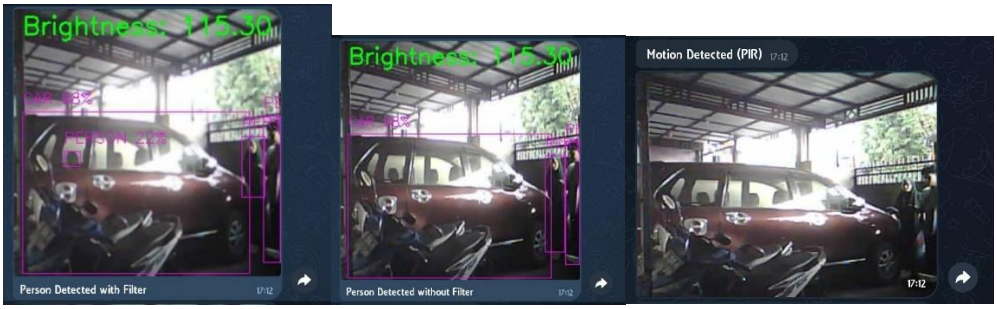
(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 1 Orang)



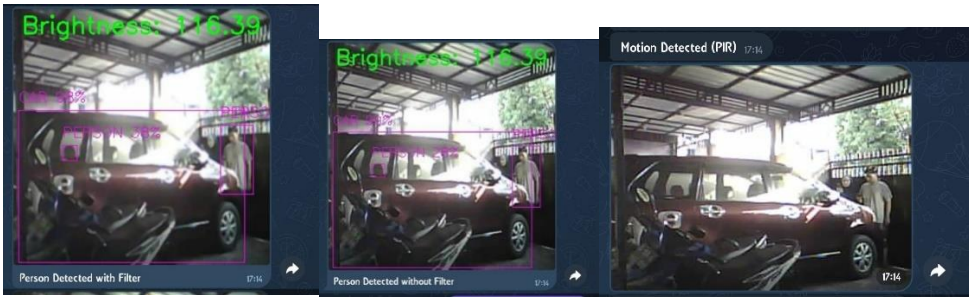
(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 2 Orang)



(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 2 Orang)

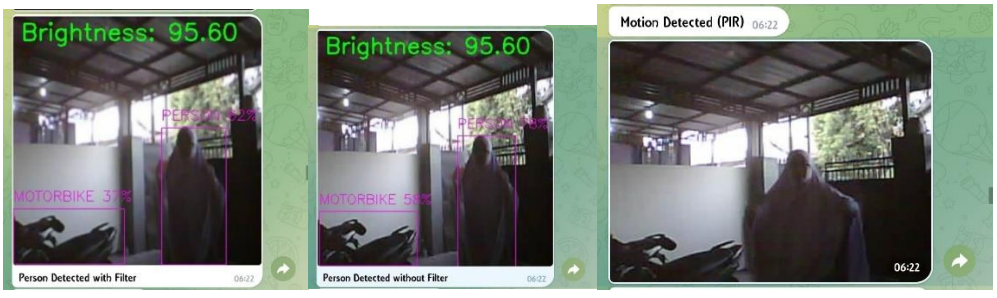


(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 2 Orang)

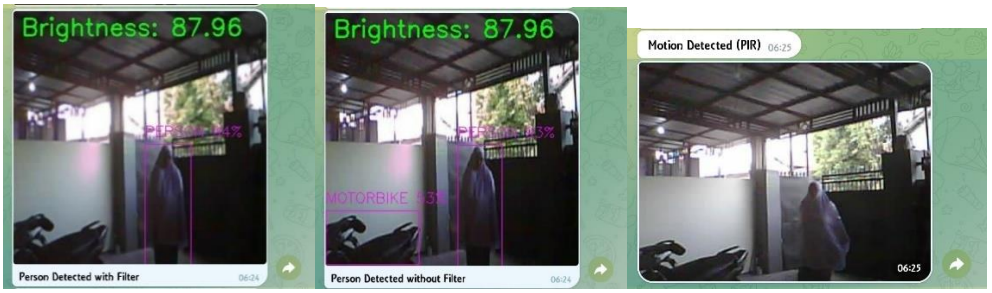


(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 2 Orang)

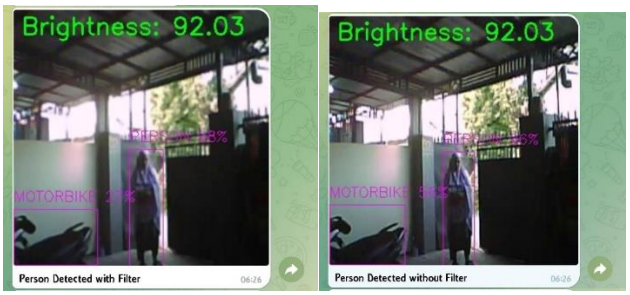
Pengujian Pemakaian Secara Real Time Diluar Ruangan (Kondisi Pagi)



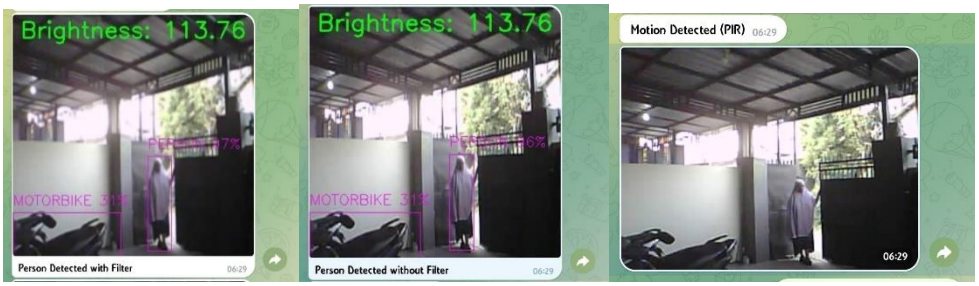
(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 1 Orang)



(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 1 Orang)

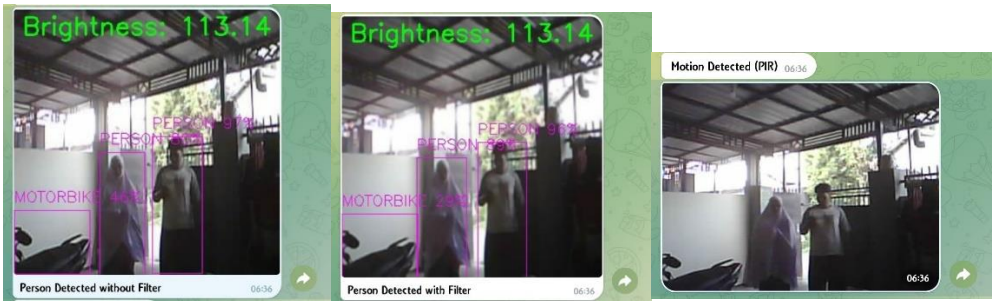


(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 1 Orang)

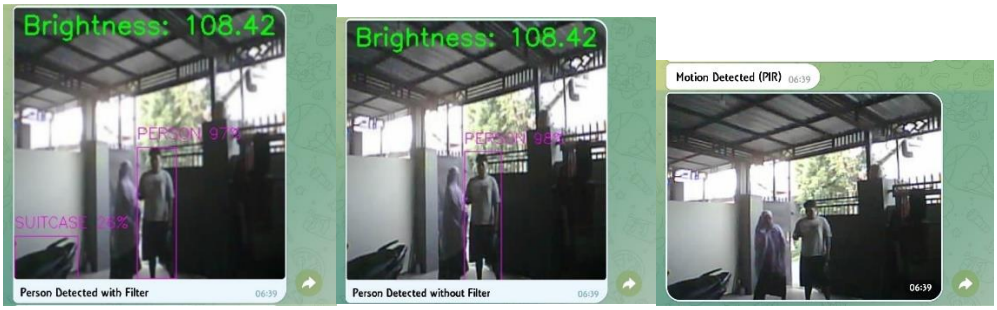


(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 1 Orang)

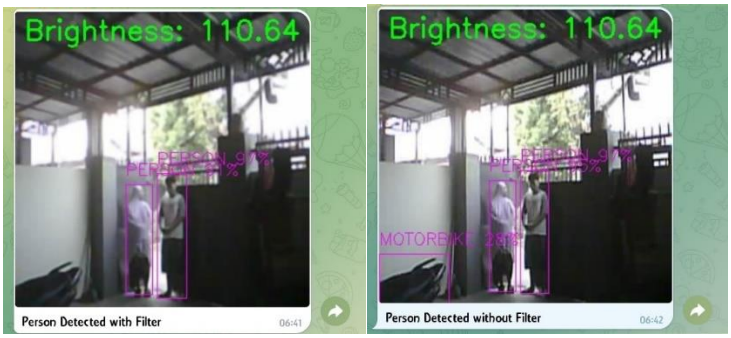
(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 2 Orang)



(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 2 Orang)



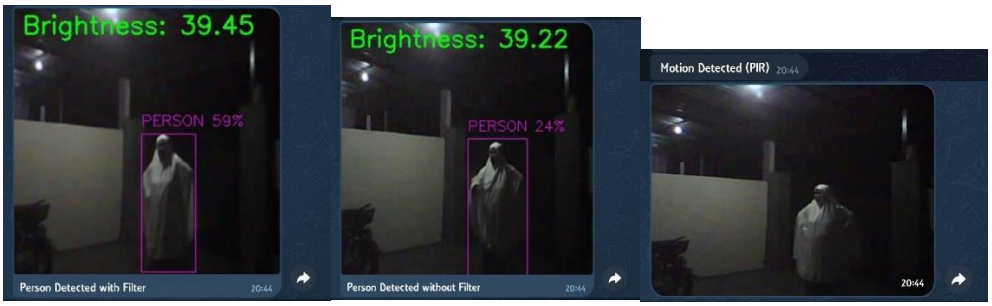
(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 2 Orang)



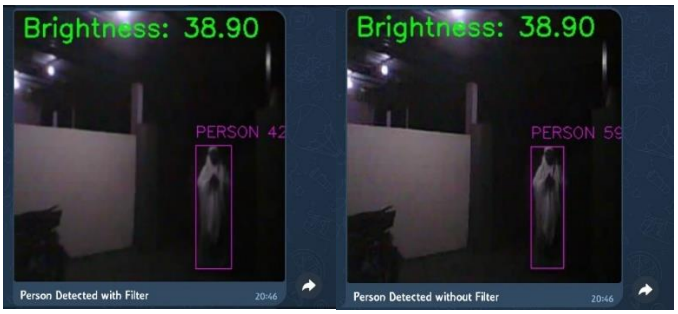
(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 2 Orang)

Pengujian Pemakaian Secara Real Time Diluar Ruangan (Kondisi Malam)

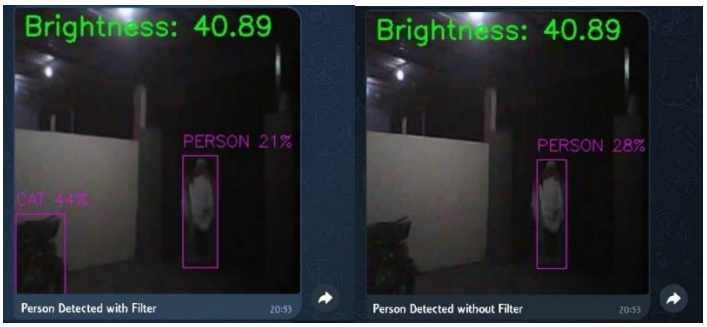
(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 1 Orang)



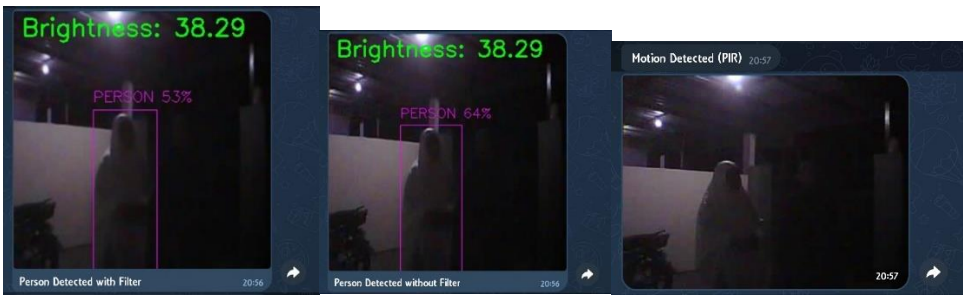
(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 1 Orang)



(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 1 Orang)

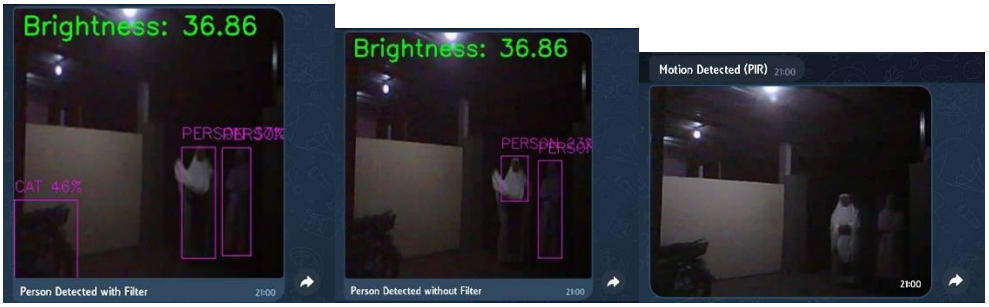


(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 1 Orang)

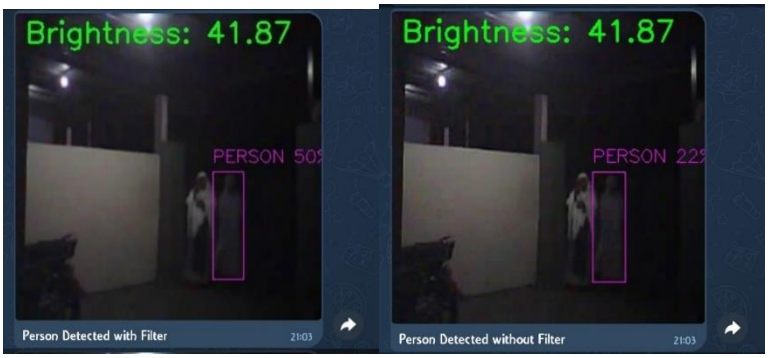


(Dok. Percobaan Jarak 2 Meter Diluar Ruangan Objek 2 Orang)

(Dok. Percobaan Jarak 4 Meter Diluar Ruangan Objek 2 Orang)



(Dok. Percobaan Jarak 6 Meter Diluar Ruangan Objek 2 Orang)



(Dok. Percobaan Jarak 7 Meter Diluar Ruangan Objek 2 Orang)