DAFTAR PUSTAKA

[1] Sumarsono, Agus. 2017. Kerusakan Jalan. Diakses pada 10 September 2021

[2] Dirjen Bina Marga, "Spesifikasi Perkerasan Aspal," Spesifikasi Umum Pekerj. Jalan dan Jemb., vol. Modul 7, pp. 1–120, 2016.

[3] Yoder, E.J. and Witczak, M.N. 1975. *Principle of Pavement Design*

[4] Kementerian PUPR, "Rencana Strategi Tahun 2020-2024," *Angew. Chemie Int. Ed.* 6(11), 951–952., pp. 1–414, 2021.

[5] Fauzan, A.A.A. and Utaminingrum, F., 2021. Sistem Pendeteksi Dini Lubang pada Jalan menggunakan *Gray Level Co-Occurrence Matrix* berbasis *Raspberry Pi*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer *e-ISSN*, *2548*, p.964X.

[6] De Zoysa, K., Keppitiyagama, C., Seneviratne, G.P. and Shihan, W.W.A.T., 2007, August. *A public transport system based sensor network for road surface condition monitoring. In Proceedings of the 2007 workshop on Networked systems for developing regions* (pp. 1-6).

[7] Shah, A., Sharma, G. and Bhargava, L., 2021, January. *Smart Implementation of Computer Vision and Machine Learningfor Pothole Detection*. *In 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 65-69). IEEE.

[8] Vigneshwar, K. and Kumar, B.H., 2016, December. *Detection and counting of pothole using image processing techniques. In 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)* (pp. 1-4). IEEE.

[9] Silvister, S., Komandur, D., Kokate, S., Khochare, A., More, U., Musale, V. and Joshi, A., 2019, December. *Deep learning approach to detect potholes in real-time using smartphone*. In *2019 IEEE Pune Section International Conference (PuneCon)* (pp. 1-4). *IEEE*.

[10] Rus, Tatag Yufitra, and Mahfud Mahfud. "*Road Damage Analysis, and Repair Techniques and Estimation Plan for Road Repair in Tani Bakti Village STA 00–03+ 00*) Kutai Kartanegara *Regency*." *Nusantara* 2.1 (2023): 27-42.

ɔs://dinaspupr.bandaacehkota.go.id/2020/06/28/berbagai-jenis-kerusakan--aspal-apa-penyebab-dan-solusinya/)

[12] Nicholls, C., Kubanek, K., Karcher, C., Hartmann, A., Adesiyun, A., Ipavec, A., Komacka, J. and Nielsen, E., 2014, April. *Durable pothole repairs*. In *transport research arena (TRA) 5th Conference: Transport Solutions from Research to Deployment. Paris, France*.

[13](https://www.carmudi.co.id/journal/penyebab-kerusakan-mobil-waspadai-lubang-di-jalan-pasca-musim-hujan/

[14]Buza, E., Omanovic, S., Huseinovic**,** 2013**.** *A Pothole Detection with Image Processing and Spectral Clustering.*

[15] J. S. Miller and W. Y. Bellinger, 2003. *Distress identification manual for the long-term pavement performance program*,

[16] Raya, J.K., 1992. *A Guide to Visual Assessment of Flexible Pavement Surface Conditions. Kuala Lumpur. IKRAM*.

[17] Tobias, D.S. and Widiarti, A.R., 2016. Deteksi glaukoma pada citra fundus retina dengan metode *K-Nearest Neighbor*. In *Seminar Nasional Ilmu Komputer (SNIK 2016)* (pp. 92-99).

[18] Eleyan, A. and Demirel, H., 2011. *Co-occurrence matrix and its statistical features as a new approach for face recognition*. *Turkish Journal of Electrical Engineering and Computer Sciences*, *19*(1), pp.97-107.

[19] Prasetyo, Eko. 2012.Data mining Konsep dan Aplikasi menggunakan Matlab, ANDI, Yogyakarta, 177

[20] Y Liu, YF Zheng. 2012. FS_SFS: *Anovel feature selection method for support vector machines, The Ohio State University, Columbus OH 43210, USA. Pattern recognition.*

[21] Santosa, B. 2007. Data Mining Teknik Pemanfaatan Data untuk Keperluan Bisnis. Graha Ilmu: Yogyakarta.

[22] Rastogi, R., Kumar, U., Kashyap, A., Jindal, S. and Pahwa, S., 2020, December. *A Comparative Evaluation of the Deep Learning Algorithms for Pothole Detection*. In *2020 IEEE 17th India Council International Conference (INDICON)* (pp. 1-6). IEEE.

llaswamy, C., Saravanan, M., Kanchana, E. and Shalini, J., 2020, July. › *learning based pothole detection and reporting system*. In *2020 7th national Conference on Smart Structures and Systems (ICSSS)* (pp. 1-6).

IEEE.

[24] Du, R., Qiu, G., Gao, K., Hu, L. and Liu, L., 2020. *Abnormal road surface recognition based on smartphone acceleration sensor. Sensors*, *20*(2), p.451.

[25] Wu, C., Wang, Z., Hu, S., Lepine, J., Na, X., Ainalis, D. and Stettler, M., 2020. *An automated machine-learning approach for road pothole detection using smartphone sensor data*. *Sensors*, *20*(19), p.5564.

[26] Hall-Beyer, M., 2017. *GLCM texture: A tutorial v. 3.0 March 2017.*

[27] Nugroho, A.S., Witarto, A.B. and Handoko, D., 2003. S*upport vector machine* teori dan aplikasinya dalam bioinformatika. *Kuliah Umum Ilmu Komputer. Com.*

LAMPIRAN

1. Source Code Program Menggunakan Blur

```
#import requirement yang digunakan pada program
import cv2
import numpy as np
from skimage.feature import graycomatrix, graycoprops
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
from skimage.feature import graycomatrix, graycoprops
import os
import xml.etree.ElementTree as ET
import matplotlib.pyplot as plt
from matplotlib import cm
import re
import pandas as pd
import joblib
import time
##########################
## LOAD MODEL YANG TELAH DI BUILD
clf2 = joblib.load("model3.pkl") #sesuaikan dengan nama modelnya
# fungsi GLCM
def calc_glcm_all_agls(img, props, dists=[5], agls=[0, np.pi/4,
np.pi/2, 3*np.pi/4], lvl=256, sym=True, norm=True):
    glcm = graycomatrix(img,
                        distances=dists,
                        angles=agls,
                        levels=lvl,
                        symmetric=sym,
                        normed=norm)
```

```
    feature = []

    glcm_props = [propery for name in props for propery in
graycoprops(glcm, name)[0]]

    for item in glcm_props:

            feature.append(item)

    return feature

properties = ['dissimilarity', 'correlation', 'homogeneity',
'contrast', 'ASM', 'energy']

# BUKA VIDIO UNTUK DI PROSES

path_file_video = 'jalan-lubang.mp4'

cap = cv2.VideoCapture(path_file_video)

frame_width = int(cap.get(3))

frame_height = int(cap.get(4))

# Definisi video codec

fourcc = cv2.VideoWriter_fourcc(*'mp4v')

out = cv2.VideoWriter('output_video_pakai_blur.mp4', fourcc, 20.0,
(frame_width, frame_height))

#hitung FPS

fps_start_time = time.time()

fps_counter = 0

# iterasi vidio yang telah di buka

while cap.isOpened():

    ret, image = cap.read()

    if not ret:

        break

    #convert ke grayscale dan Blur

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

    gray2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    ;i contour edgenya pake canny

    ;d = cv2.Canny(blurred, 0, 255) #low_threshold,
    ;eshold
```

```
    #cari contour

    contours,  hierarchy = cv2.findContours(edged.copy(),
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    bounding = []

    try: hierarchy = hierarchy[0]

    except: hierarchy = []

    height, width = edged.shape

    min_x, min_y = width, height

    max_x = max_y = 0

    # Hitung bounding box dan tampilkan di frame

    for contour, hier in zip(contours, hierarchy):

        (x,y,w,h) = cv2.boundingRect(contour)

        min_x, max_x = min(x, min_x), max(x+w, max_x)

        min_y, max_y = min(y, min_y), max(y+h, max_y)

        if w > 80 and h > 80:

            cv2.rectangle(image, (x,y), (x+w,y+h), (0, 0, 255), 2)
#bikin rectangle warna merah

            bounding.append([(x,y), (x+w,y+h)]) #ambil bounding
kecil #kasi uncomment ini kalau mau prediksi semua yang kecil-
kecil juga

    if max_x - min_x > 0 and max_y - min_y > 0:

        cv2.rectangle(image, (min_x, min_y), (max_x, max_y), (0,
0, 255), 2) #set warna merah

        bounding.append([ (min_x, min_y), (max_x, max_y)]) #ambil
bounding besar

    print("bounding coords")

    print(bounding)

    allCropped = []

    for c in bounding:

      print("crop:"+str(c[0][0])+","+str(c[0][1])+" |
"+str(c[1][0])+","+str(c[1][1]))

        x = c[0][0]

          = c[0][1]

          = c[1][1]-c[0][1];

          = c[1][0]-c[0][0];
```

```
        print("h:"+str(h))
        print("w:"+str(w))
        cropped_image = gray2[y:y+h,x:x+w]
        allCropped.append(cropped_image)

    properties = ['dissimilarity', 'correlation', 'homogeneity',
'contrast', 'ASM', 'energy']

    glcm_all_agls = []
    for img in allCropped:

glcm_all_agls.append(calc_glcm_all_agls(img,props=properties))
        columns = []
        angles = ['0', '45', '90','135']
        for name in properties :
            for ang in angles:
                columns.append(name + "_" + ang)

     # bikin df buat GLCM features data.
    #ini nanti yang dipake buat inference
    glcm_df = pd.DataFrame(glcm_all_agls,
                           columns = columns)
    ## PREDIKSI ##
    #ini prediksi  pake model yang di training
    predictions = clf2.predict(glcm_df)
    print(predictions)
    # Define the color for the text (you can adjust this)
    text_color = (0, 0, 255)  # Red color in BGR
    # image
        :erate through each bounding box and label
         box, label in zip(bounding, predictions):
         (start_x, start_y), (end_x, end_y) = box
```

```
        # Draw the bounding box

        cv2.rectangle(image, (start_x, start_y), (end_x, end_y),
(0, 255, 0), 2)  # Green color for bounding boxes



        # Define the position for the label text (you can adjust
this)

        text_x = start_x

        text_y = start_y - 10  # Place the label just above the
bounding box



        # Put the label text on the image

        cv2.putText(image, label, (text_x, text_y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, text_color, 2)

    fps_counter += 1



    fps = fps_counter / (time.time() - fps_start_time)

    fps_text = f"FPS : {fps:.2f}"

    text_size = cv2.getTextSize(fps_text,
cv2.FONT_HERSHEY_SIMPLEX,1,2)[0]

    text_x = frame_width - text_size[0] - 10

    text_y = frame_height - 30

    cv2.putText(image, fps_text, (text_x, text_y),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

    fps_counter = 0

    fps_start_time = time.time()



    # TAMPILKAN PROSES PREDIKSI

    cv2.namedWindow("result", cv2.WINDOW_AUTOSIZE)

    cv2.imshow("result", image)

    if cv2.waitKey(1) & 0xFF == ord("q"):

        break

    [MPAN VIDIO KE DIRECTORY

    .write(image)

    se video objects
```

```
cap.release()

out.release()

# from IPython.display import Video

# # Path to your output video file

# output_video_path = 'output_video.avi'

# # Display the video

# Video(output_video_path)
```

2. Source Code Program Tanpa Blur

```
#import requirement yang digunakan pada program

import cv2

import numpy as np

from skimage.feature import graycomatrix, graycoprops

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, confusion_matrix

from skimage.feature import graycomatrix, graycoprops

import os

import xml.etree.ElementTree as ET

import matplotlib.pyplot as plt

from matplotlib import cm

import re

import pandas as pd

import joblib


##########################

## LOAD MODEL YANG TELAH DI BUILD

joblib.load("model-no-blur.pkl") #sesuaikan dengan nama
a
```

```
# fungsi GLCM

def calc_glcm_all_agls(img, props, dists=[5], agls=[0, np.pi/4,
np.pi/2, 3*np.pi/4], lvl=256, sym=True, norm=True):


    glcm = graycomatrix(img,

                        distances=dists,

                        angles=agls,

                        levels=lvl,

                        symmetric=sym,

                        normed=norm)

    feature = []

    glcm_props = [propery for name in props for propery in
graycoprops(glcm, name)[0]]

    for item in glcm_props:

            feature.append(item)


    return feature


properties = ['dissimilarity', 'correlation', 'homogeneity',
'contrast', 'ASM', 'energy']


# BUKA VIDIO UNTUK DI PROSES

path_file_video = 'jalan-lubang.mp4'

cap = cv2.VideoCapture(path_file_video)

frame_width = int(cap.get(3))

frame_height = int(cap.get(4))

# Definisi video codec

fourcc = cv2.VideoWriter_fourcc(*'mp4v')

out = cv2.VideoWriter('output_video_tanpa_blur.mp4', fourcc, 20.0,
      vidth, frame_height))

      si vidio yang telah di buka

      ap.isOpened():
```

```
    ret, image = cap.read()
    if not ret:
        break

    #convert ke grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edged = cv2.Canny(gray2, 0, 255) #low_threshold,
high_threshold

    #cari contour
    contours,  hierarchy = cv2.findContours(edged.copy(),
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    bounding = []
    try: hierarchy = hierarchy[0]
    except: hierarchy = []

    height, width = edged.shape
    min_x, min_y = width, height
    max_x = max_y = 0

    # Hitung bounding box dan tampilkan di frame
    for contour, hier in zip(contours, hierarchy):
        (x,y,w,h) = cv2.boundingRect(contour)
        min_x, max_x = min(x, min_x), max(x+w, max_x)
        min_y, max_y = min(y, min_y), max(y+h, max_y)
        if w > 80 and h > 80:
            cv2.rectangle(image, (x,y), (x+w,y+h), (0, 0, 255), 2)
#bikin rectangle warna merah
            bounding.append([(x,y), (x+w,y+h)]) #ambil bounding
kasi uncomment ini kalau mau prediksi semua yang kecil-
nga


max_x - min_x > 0 and max_y - min_y > 0:
```

```
        cv2.rectangle(image, (min_x, min_y), (max_x, max_y), (0,
0, 255), 2) #set warna merah

        bounding.append([ (min_x, min_y), (max_x, max_y)]) #ambil
bounding besar


    print("bounding coords")

    print(bounding)


    allCropped = []

    for c in bounding:

      print("crop:"+str(c[0][0])+","+str(c[0][1])+" |
"+str(c[1][0])+","+str(c[1][1]))

      x = c[0][0]

      y = c[0][1]

      h = c[1][1]-c[0][1];

      w = c[1][0]-c[0][0];

      print("h:"+str(h))

      print("w:"+str(w))

      cropped_image = gray2[y:y+h,x:x+w]

      allCropped.append(cropped_image)


    properties = ['dissimilarity', 'correlation', 'homogeneity',
'contrast', 'ASM', 'energy']


    glcm_all_agls = []

    for img in allCropped:


glcm_all_agls.append(calc_glcm_all_agls(img,props=properties))

      columns = []

      angles = ['0', '45', '90','135']

      for name in properties :

          for ang in angles:

              columns.append(name + "_" + ang)
```

```
 # bikin df buat GLCM features data.
#ini nanti yang dipake buat inference
glcm_df = pd.DataFrame(glcm_all_agls,
                       columns = columns)


## PREDIKSI ##
#ini prediksi  pake model yang di training
predictions = clf2.predict(glcm_df)
print(predictions)


# Define the color for the text (you can adjust this)
text_color = (0, 0, 255)  # Red color in BGR
# image
# Iterate through each bounding box and label
for box, label in zip(bounding, predictions):
    (start_x, start_y), (end_x, end_y) = box


    # Draw the bounding box
    cv2.rectangle(image, (start_x, start_y), (end_x, end_y),
(0, 255, 0), 2)  # Green color for bounding boxes


    # Define the position for the label text (you can adjust
this)
    text_x = start_x
    text_y = start_y - 10  # Place the label just above the
bounding box


    # Put the label text on the image
    cv2.putText(image, label, (text_x, text_y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, text_color, 2)


AMPILKAN PROSES PREDIKSI
.namedWindow("result", cv2.WINDOW_AUTOSIZE)
```

```
        cv2.imshow("result", image)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

        # SIMPAN VIDIO KE DIRECTORY
        out.write(image)

# Release video objects
cap.release()
out.release()

# from IPython.display import Video

# # Path to your output video file
# output_video_path = 'output_video.avi'

# # Display the video
# Video(output_video_path)
```