APPENDICES

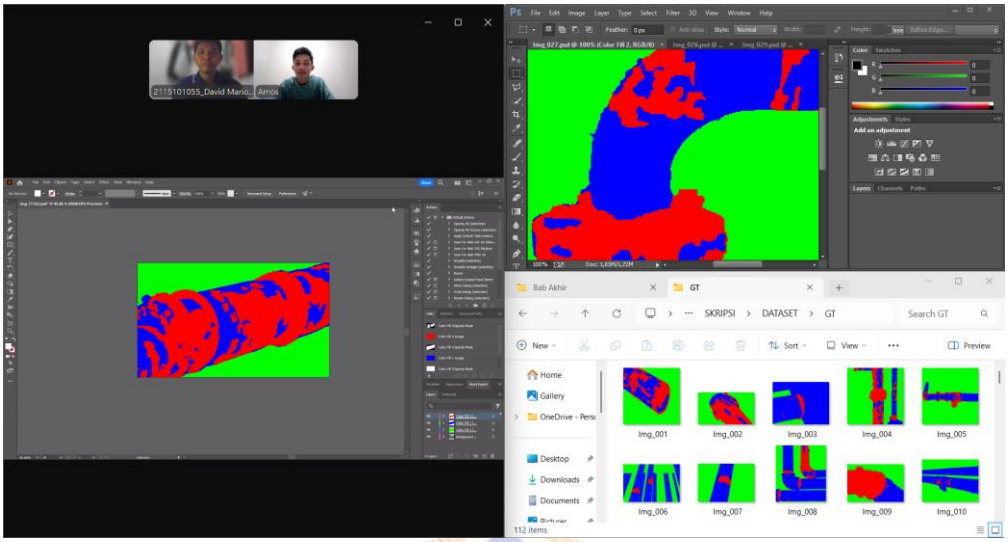
Appendix 1. Identity of Research Assistant on Image Data

1st Labeller and Expert

Name	Eduardus Aditya Rian Prasetyo
Gender	Male
Graduate	D3 Heavy Equipment Engineering
Profession	Diesel Powerplant Engineer
Workplace Agency	PT Freeport Indonesia

2nd Labeller

Name	Amos Alexandro Lumban Tobing
Gender	Male
Graduate	S1 Mechanical Engineering
Profession	Crew at Pump House Mechanic
Workplace Agency	PT Indonesia Morowali Industrial Park



Appendix 2 Code of Preparation

```
if not is_mask:
   # Preprocess original images
   img = img.convert("RGB")
   img = img.resize(target, Image.Resampling.LANCZOS)
   img = normalize_image(img)
else:
# Preprocess ground truth images (masks)
   img = img.convert("RGB")
   img = img.resize(target, Image.Resampling.NEAREST)
   img = enhance_contrast_for_mask(img)
   img = threshold_colors(img)

# Save the processed image with the modified filename and
change the extension to PNG
new_filename = os.path.splitext(filename)[0] + '.png'  # Change
the file extension to .png
img.save(os.path.join(dst_folder, new_filename), format='PNG')
```

Appendix 3 Code of Augmentation

- Rotate

```
def rotate_and_save(image, angle, original_name,
extension, output_folder):
    rotated_image = image.rotate(angle)
    new_filename =
f"{original_name}_rotated_{angle}{extension}"
    rotated_image.save(os.path.join(output_folder,
new_filename))

for angle in [15, 30, 45, 60, 90]:
      rotate_and_save(img, angle, file_name,
file_extension, rotate_ori)
      rotate_and_save(gt, angle,
file_name.replace('ORI_', 'GT_'), file_extension,
rotate_gt)
```

- Flip

```
def flip_and_save(image, flip_type, original_name,
extension, output_folder):
      if flip_type == 'horizontal':
            flipped_image =
      image.transpose(Image.FLIP_LEFT_RIGHT)
      elif flip_type == 'vertical':
            flipped_image =
      image.transpose(Image.FLIP_TOP_BOTTOM}
      new_filename =
f"{original_name}_flipped_{flip_type}{extension}"
      flipped_image.save(os.path.join(output_folder,
new_filename))
```

- Hue

```
def change_hue(image, hue_shift):
    img_hsv = image.convert('HSV')
    np_img_hsv = np.array(img_hsv, dtype=np.uint8)
    hue_channel = np_img_hsv[..., 0].astype(np.int16)
    hue_channel = (hue_channel + hue_shift) % 256
    np_img_hsv[..., 0] = hue_channel.astype(np.uint8)
    img_rgb = Image.fromarray(np_img_hsv,
'HSV').convert('RGB')
    return img_rgb
```

- Saturation

```
def change_saturation(image, saturation_factor):
    enhancer = ImageEnhance.Color(image)
    return enhancer.enhance(saturation_factor)
```

- Brightness

```
def change_brightness(image, brightness_factor):
    enhancer = ImageEnhance.Brightness(image)
    return enhancer.enhance(brightness_factor)
```

- Blur

```
blur_radii = [1, 2, 3]

for radius in blur_radii:
        augmented_image_blur =
apply_gaussian_blur(Image.open(image_path), radius)
        blur_name = f"{image_name}_blur_{radius}.png"  #
Output always .png
        augmented_image_blur.save(os.path.join(blur_ori,
blur_name))

        # Copy ground truth image with the corresponding
name, with "_mask" format
        gt_augmented_name =
f"GT_{image_name[4:]}_blur_{radius}.png"  # Output mask
always .png
        shutil.copy(gt_image_path, os.path.join(blur_gt,
gt_augmented_name))
```

Appendix 4 Code of Split Data

```
if (train_ratio + val_ratio + test_ratio) != 1.0:
    raise ValueError("Rasio train, validation, dan test harus
berjumlah 1.0")

train_original, temp_original, train_ground_truth,
temp_ground_truth = train_test_split(
    original_files, ground_truth_files, test_size=(1.0 -
train_ratio), random_state=42)

val_size = val_ratio / (val_ratio + test_ratio)

val_original, test_original, val_ground_truth,
test_ground_truth = train_test_split(
    temp_original, temp_ground_truth, test_size=(1.0 -
val_size), random_state=42)
```

Appendix 5 Code of One Hot Encode

```
def encode_mask(mask_image, tolerance=3):

    mask_image[np.all(mask_image == [0, 0, 0], axis=-1)] = [0,
255, 0]

    encoded_mask = np.zeros((mask_image.shape[0],
mask_image.shape[1]), dtype=np.uint8)

    red_mask = np.all(np.abs(mask_image - [255, 0, 0]) <=
tolerance, axis=-1)
    encoded_mask[red_mask] = 0

    green_mask = np.all(np.abs(mask_image - [0, 255, 0]) <=
tolerance, axis=-1)
    encoded_mask[green_mask] = 2

    blue_mask = np.all(np.abs(mask_image - [0, 0, 255]) <=
tolerance, axis=-1)
    encoded_mask[blue_mask] = 1

    return encoded_mask
```

Appendix 6 Code of Implementation of EfficientNetB1

- BiSeNetV3

```
base_model = EfficientNetB1(weights='imagenet',
include_top=False, input_tensor=input_tensor)
```

- Mobile U-Net

```
efficientnet_base = EfficientNetB1(include_top=False,
weights='imagenet',input_shape=input_size)
```

Appendix 7 Code of Training Model

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
mode='min', restore_best_weights=False, verbose=1)

model_checkpoint = ModelCheckpoint(
filepath='.//model//256A_B8E100_Mobile U-Net-
EfficientNetB1.keras', monitor='val_loss', mode='min',
save_best_only=False, verbose=1)

reduce_lr = ReduceLROnPlateau( monitor='val_loss', factor=0.1,
patience=5, mode='min', min_lr=1e-6, verbose=1)

log_dir = "logs/fit/" + datetime.now().strftime("%Y%m%d-
%H%M%S")
tensorboard = tf.keras.callbacks.TensorBoard(
    log_dir=log_dir,
    histogram_freq=1,
    write_graph=True,
    write_images=True
)

callbacks = [early_stopping, model_checkpoint, reduce_lr,
tensorboard]

start_time = time.time()

model.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train,
                    epochs=100,
                    batch_size=8,
                    validation_data=(X_val, y_val),
                    callbacks = callbacks)

end_time = time.time()
training_time = end_time - start_time
```

Appendix 8 Mobile U-Net Metric Results against Original Test Data

Image	Value of Metrics Evaluation					Time
	Metrics	**Class 0**	**Class 1**	**Class 2**	**Average**	
Image 1	mIoU	0,081	0,4628	0,9159	0,4866	0,3882
	Dice	0,7563	0,6328	0,9561	0,7817	
Image 2	mIoU	0,4096	0,1565	0,4027	0,3229	0,3695
	Dice	0,5812	0,2707	0,5742	0,4754	
Image 3	mIoU	0,783	0,6094	0,9809	0,7911	0,3547
	Dice	0,8783	0,7573	0,9904	0,8753	
Image 4	mIoU	0,6914	0,6661	0,893	0,7502	0,3883

Image	Value of Metrics Evaluation				Time
	Metrics	Class 0	Class 1	Class 2	Average	
	Dice	0,8175	0,7996	0,9435	0,8535	
Image 5	mIoU	0,7425	0,7361	0,7351	0,7379	0,4193
	Dice	0,8522	0,848	0,8473	0,8492	
Image 6	mIoU	0,5947	0,7605	0,8413	0,7322	0,4072
	Dice	0,7459	0,864	0,9138	0,8412	
Image 7	mIoU	0,8457	0,8016	0,9707	0,8727	0,3757
	Dice	0,9164	0,8899	0,9851	0,9305	
Image 8	mIoU	0,4318	0,4224	0,2165	0,3569	0,378
	Dice	0,6031	0,5939	0,3559	0,5176	
Image 9	mIoU	0,1602	0,9005	0,9674	0,6760	0,3618
	Dice	0,2762	0,9476	0,9834	0,7357	
Image 10	mIoU	0,7548	0,5758	0,931	0,7539	0,4664
	Dice	0,8603	0,8622	0,9642	0,8956	
Image 11	mIoU	0,6396	0,6338	0,745	0,6728	0,3809
	Dice	0,7802	0,7759	0,8538	0,8033	
Image 12	mIoU	0,5073	0,3843	0,4289	0,4402	0,655
	Dice	0,6731	0,5553	0,6003	0,6096	
Average of Computation Time						0,4121

Appendix 9 Mobile U-Net Metric Results against Augmented Test Data

Image	Value of Metrics Evaluation				Time
	Metrics	Class 0	Class 1	Class 2	Average	
Image 1	mIoU	0,9288	0,9286	0,9959	0,9511	0,376
	Dice	0,9631	0,963	0,9979	0,9747	
Image 2	mIoU	0,8422	0,9192	0,9951	0,9188	0,3579
	Dice	0,9143	0,9579	0,9975	0,9566	
Image 3	mIoU	0,9745	0,9461	0,9959	0,9722	0,364
	Dice	0,9871	0,9732	0,998	0,9861	
Image 4	mIoU	0,9205	0,948	0,9919	0,9535	0,3729
	Dice	0,9586	0,9733	0,9959	0,9759	
Image 5	mIoU	0,966	0,9881	0,9896	0,9812	0,3699
	Dice	0,9827	0,994	0,9948	0,9905	
Image 6	mIoU	0,9689	0,9738	0,9936	0,9788	0,3645
	Dice	0,9842	0,9867	0,9968	0,9892	
Image 7	mIoU	0,9385	0,9115	0,9917	0,9472	0,3682
	Dice	0,9683	0,9537	0,9958	0,9726	

Image	Value of Metrics Evaluation					Time
	Metrics	Class 0	Class 1	Class 2	Average	
Image 8	mIoU	0,9679	0,9888	0,9879	0,9815	0,3776
	Dice	0,9837	0,9944	0,9939	0,9907	
Image 9	mIoU	0,7394	0,9746	0,9919	0,9020	0,3717
	Dice	0,8502	0,9871	0,9959	0,9444	
Image 10	mIoU	0,896	0,8956	0,9766	0,9227	0,4546
	Dice	0,9451	0,9449	0,9882	0,9594	
Image 11	mIoU	0,9199	0,9249	0,954	0,9329	0,6651
	Dice	0,9583	0,961	0,9765	0,9653	
Image 12	mIoU	0,757	0,5206	0,6625	0,6467	0,3698
	Dice	0,8617	0,6847	0,797	0,7811	
Average of Computation Time						0,4010

Appendix 10 BiSeNetV3 Metric Results against Original Test Data

Image	Value of Metrics Evaluation					Time
	Metrics	Class 0	Class 1	Class 2	Average	
Image 1	mIoU	0,5085	0,2838	0,958	0,5834	0,5397
	Dice	0,6742	0,4421	0,9785	0,6983	
Image 2	mIoU	0,4583	0,1673	0,5166	0,3807	0,4891
	Dice	0,6286	0,2866	0,6813	0,5322	
Image 3	mIoU	0,4188	0,3755	0,9743	0,5895	0,6481
	Dice	0,5903	0,546	0,987	0,7078	
Image 4	mIoU	0,6515	0,6702	0,9212	0,7476	0,4471
	Dice	0,789	0,8025	0,959	0,8502	
Image 5	mIoU	0,7807	0,7559	0,7813	0,7726	0,4704
	Dice	0,8768	0,861	0,8772	0,8717	
Image 6	mIoU	0,556	0,6666	0,8102	0,6776	0,4773
	Dice	0,7146	0,8	0,8952	0,8033	
Image 7	mIoU	0,7881	0,7043	0,9611	0,8178	0,4662
	Dice	0,8815	0,8265	0,9802	0,8961	
Image 8	mIoU	0,4037	0,454	0,3969	0,4182	0,4846
	Dice	0,5752	0,6244	0,5682	0,5893	
Image 9	mIoU	0	0,8988	0,9586	0,6191	0,5023
	Dice	0	0,9467	0,9789	0,6419	
Image 10	mIoU	0,6758	0,69	0,9119	0,7592	0,5341
	Dice	0,8065	0,8166	0,9539	0,8590	
Image 11	mIoU	0,6884	0,7345	0,781	0,7346	0,4606

Image	Value of Metrics Evaluation					Time
	Metrics	Class 0	Class 1	Class 2	Average	
	Dice	0,8154	0,8469	0,877	0,8464	
Image 12	mIoU	0,507	0,3994	0,4652	0,4572	0,5517
	Dice	0,6729	0,5709	0,635	0,6263	
Average of Computation Time						0,5059

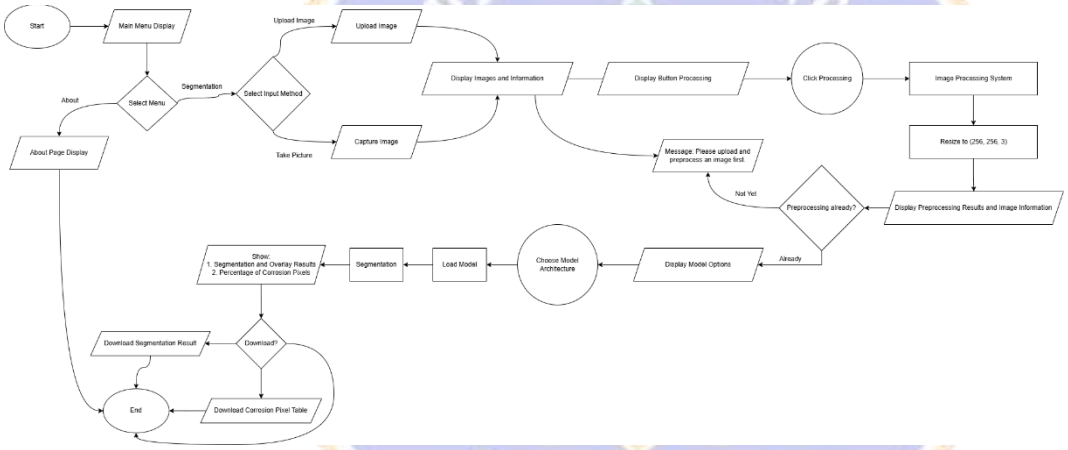
Appendix 11 BiSeNetV3 Metric Results against Augmented Test Data

Image	Value of Metrics Evaluation					Time
	Metrics	Class 0	Class 1	Class 2	Average	
Image 1	mIoU	0,9705	0,9697	0,9978	0,9793	0,4721
	Dice	0,985	0,9846	0,9989	0,9895	
Image 2	mIoU	0,9253	0,9612	0,9979	0,9615	0,5051
	Dice	0,9612	0,9802	0,9909	0,9774	
Image 3	mIoU	0,9887	0,9746	0,9985	0,9873	0,4261
	Dice	0,9943	0,9872	0,9992	0,9936	
Image 4	mIoU	0,9697	0,9797	0,9969	0,9821	0,4892
	Dice	0,9846	0,9897	0,9984	0,9909	
Image 5	mIoU	0,9818	0,9934	0,9952	0,9901	0,4301
	Dice	0,9908	0,9967	0,9976	0,9950	
Image 6	mIoU	0,989	0,99	0,9976	0,9922	0,4761
	Dice	0,9945	0,995	0,9988	0,9961	
Image 7	mIoU	0,9639	0,9463	0,9964	0,9689	0,5709
	Dice	0,9816	0,9724	0,9982	0,9841	
Image 8	mIoU	0,9825	0,9942	0,9945	0,9904	0,4657
	Dice	0,9912	0,9971	0,9972	0,9952	
Image 9	mIoU	0,8682	0,9886	0,997	0,9513	0,47
	Dice	0,9294	0,9943	0,9985	0,9741	
Image 10	mIoU	0,956	0,9549	0,99	0,9670	0,4679
	Dice	0,9775	0,9769	0,995	0,9831	
Image 11	mIoU	0,9749	0,9761	0,9864	0,9791	0,4357
	Dice	0,9873	0,9879	0,9931	0,9894	
Image 12	mIoU	0,7931	0,7006	0,8623	0,7853	0,4976
	Dice	0,9718	0,9738	0,9917	0,9791	
Average of Computation Time						0,4755

Appendix 12 Model Deployment to Website

Repository	:	https://github.com/055DavidMario/Thesis-Project-Corrosion-Pipes-Image-Segmentation.git
Website	:	https://dmys-cpis.streamlit.app/

Appendix 13 User Flowchart for Using Web Streamlit Segmentation



BIOGRAPHY

David Mario Yohanes Samosir was born in Cibinong Bogor on 06 September 2002. The author is an Indonesian citizen and a Catholic. The author currently lives in Ciriung Village, Cibinong District, Bogor Regency. The author completed his primary education at the Eka Wijaya Cibinong Foundation and graduated in 2014. Then, the author continued his education at Yayasan Eka Wijaya Cibinong and graduated in 2017. In 2020, the author graduated from SMK Negeri 1 Cibinong. The author is registered as a student of the Computer Science Undergraduate Study Program at Universitas Pendidikan Ganesha from 2021 until the writing of this undergraduate thesis.