Daftar Pustaka

[1] A. Ahmad, "Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning," *J. Teknol. Indones.*, no. October, p. 3, 2017.

[2] C. Purnamaningsih, R. Saptono, and A. Aziz, "Pemanfaatan Metode K-Means Clustering dalam Penentuan Penjurusan Siswa SMA," *J. Teknol. Inf. ITSmart*, vol. 3, no. 1, p. 27, 2016.

[3] S. Hadianti and D. Riana, "Segmentasi Citra Bemisia Tabaci Menggunakan Metode K-Means," *Semin. Nas. Inov. dan Tren*, p. 2018, 2018.

[4] H. Eum, J. Bae, C. Yoon, and E. Kim, "Ship Detection Using Edge-Based Segmentation and Histogram of Oriented Gradient with Ship Size Ratio," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 15, no. 4, pp. 251–259, 2016.

[5] S. Desmanto, Irwan, and R. Angreni, "Penerapan Algoritma K-Means Clustering Untuk Pengelompokkan Citra Digital Dengan Ekstraksi Fitur Warna RGB," *J. Inform. dan Apl.*, vol. 1, no. x, pp. 1–9, 2015.

[6] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm," *Procedia Comput. Sci.*, vol. 54, pp. 764–771, 2015.

[7] M. Riadi, "Pengolahan Citra Digital - KajianPustaka.com," 2016. [Online]. Available: https://www.kajianpustaka.com/2016/04/pengolahan-citra-digital.html. [Accessed: 02-Aug-2019].

[8] G. Seif, "Clustering Algorithms," 2018. [Online]. Available:

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68. [Accessed: 02-Aug-2019].

[9] dkk Nur Ridha Apriyanti, "Algortima K-Means Clustering Dalam Pengolahan Citra Digital Landsat," *Ilmu Komput.*, vol. 02, no. Clustering K-Means, pp. 1–13, 2015.

[10] J. Morgan, "Differences Between Supervised Learning and Unsupervised Learning | Difference Between | Supervised Learning vs Unsupervised Learning," 2018. [Online]. Available: http://www.differencebetween.net/technology/differences-between-supervised-learning-and-unsupervised-learning/. [Accessed: 06-Mar-2019].

[11] Silontong, "Pengertian Transportasi Online, Laut, Darat dan Udara," 2018. [Online]. Available: https://www.silontong.com/2018/02/28/pengertian-transportasi/. [Accessed: 15-Mar-2019].

[12] Silontong, "Alat Transportasi Laut Tradisional dan Modern," 2017. [Online]. Available: https://www.silontong.com/2017/12/07/alat-transportasi-laut-tradisional-modern/. [Accessed: 15-Mar-2019].

[13] A. S. R. Sinaga;, "Implementasi Teknik Thresholding pada Segmentasi Citra Digital," *Mantik Penusa*, vol. 1, no. 2, p. 49, 2017.

[14] doavers, "Apa Itu Ekstraksi Fitur pada Citra Digital," 2018. [Online]. Available: https://www.doavers.com/blog/apa-itu-ekstraksi-fitur-pada-citra-digital. [Accessed: 16-Jul-2019].

[15] D. Indra, “Pendeteksian tepi objek menggunakan metode gradien,” *J. Ilm. Ilk. Vol. 8 Nomor 2*, vol. 8, no. Agustus, pp. 69–75, 2016.

LAMPIRAN

Lampiran 1 : *Source code* Program deteksi tepi menggunakan metode *prewitt* dan *sobel*

```
#Perintah untuk menjalankan hasil deteksi tepi menggunakan
cmd

#C:\Users\Firdaus>cd C:\Skripsi program

#C:\Skripsi program>pythonprewitt.py


import cv2

import numpy as np


img = cv2.imread('tinggi/kc2.jpg')

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

img_gaussian = cv2.GaussianBlur(gray,(3,3),0)



#Operator sobel

img_sobelx = cv2.Sobel(img_gaussian,cv2.CV_8U,1,0,ksize=5)

img_sobely = cv2.Sobel(img_gaussian,cv2.CV_8U,0,1,ksize=5)

img_sobel = img_sobelx + img_sobely



#Operator prewitt

kernelx = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])

kernely = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])

img_prewittx = cv2.filter2D(img_gaussian, -1, kernelx)

img_prewitty = cv2.filter2D(img_gaussian, -1, kernely)



cv2.imshow("input citra kapal", img)

cv2.imshow("Sobel X", img_sobelx)
```

```
cv2.imshow("Sobel Y", img_sobely)
cv2.imshow("Operator Sobel", img_sobel)
cv2.imshow("Prewitt X", img_prewittx)
cv2.imshow("Prewitt Y", img_prewitty)
cv2.imshow("Operator Prewitt", img_prewittx + img_prewitty)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

Lampiran 2 : *Source code* segementasi citra kapal menggunakan K-Means

```
#untuk menjalankan hasil segmentasi menggunakan cmd di bawah
ini

#C:\Users\Firdaus\cd C:\skripsi program

#C:\skripsi program>python segmentasi-using-kmeans.py -i
kargo/kargo1.jpg -w 300 -s hsv -c 02 -n 3 -o -f jpg


import numpy as np

from sklearn.cluster import KMeans

import argparse

import cv2

import datetime


ap = argparse.ArgumentParser()

ap.add_argument('-i', '--image', required=True, help='Path
to image file')

ap.add_argument('-w', '--width', type=int, default=0,

    help='Width to resize image to in pixels')

ap.add_argument('-s', '--color-space', type=str,
default='bgr',

    help='Color space to use: BGR (default), HSV, Lab, YCrCb
(YCC)')

ap.add_argument('-c', '--channels', type=str, default='all',

    help='Channel indices to use for clustering, where 0 is
the first channel,'

    + ' 1 is the second channel, etc. E.g., if BGR color
space is used, "02" '

    + 'selects channels B and R. (default "all")')

ap.add_argument('-n', '--num-clusters', type=int, default=3,

    help='Number of clusters for K-means clustering (default
3, min 2).')

ap.add_argument('-o', '--output-file', action='store_true',

    help='Save output image (side-by-side comparison of
original image and'
```

```
    + ' clustering result) to disk.')
ap.add_argument('-f', '--output-format', type=str,
default='jpg',
    help='File extension for output image (default jpg)')

args = vars(ap.parse_args())
image = cv2.imread(args['image'])

# Ubah ukuran gambar dan buat salinan dari gambar asli
if args['width'] > 0:
    height = int((args['width'] / image.shape[1]) *
image.shape[0])
    image = cv2.resize(image, (args['width'], height),
        interpolation=cv2.INTER_AREA)
orig = image.copy()

# Ubah ruang warna gambar
colorSpace = args['color_space'].lower()
if colorSpace == 'hsv':
    image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
elif colorSpace == 'ycrcb' or colorSpace == 'ycc':
    image = cv2.cvtColor(image, cv2.COLOR_BGR2YCrCb)
elif colorSpace == 'lab':
    image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
else:
    colorSpace = 'bgr'  # set for file naming purposes

# Simpan hanya saluran yang dipilih untuk pengelompokan K-
means.
if args['channels'] != 'all':
    channels = cv2.split(image)
    channelIndices = []
    for char in args['channels']:
```

```
            channelIndices.append(int(char))
        image = image[:,:,channelIndices]
        if len(image.shape) == 2:
            image.reshape(image.shape[0], image.shape[1], 1)


# meratakan gambar 2D ke dalam fitur MxN, di mana M berada
# jumlah piksel dan N adalah dimensi
reshaped = image.reshape(image.shape[0] * image.shape[1],
image.shape[2])


# Performa klasterisasi K-means .
if args['num_clusters'] < 2:
    print('Warning: num-clusters < 2 invalid. Using num-
clusters = 2')
numClusters = max(2, args['num_clusters'])
kmeans = KMeans(n_clusters=numClusters, n_init=40,
max_iter=500).fit(reshaped)


# Membentuk kembali hasil menjadi array 2D, dimana setiap
elemen mewakili indeks
# klaster piksel yang sesuai (0 hingga K-1).
clustering = np.reshape(np.array(kmeans.labels_,
dtype=np.uint8),
    (image.shape[0], image.shape[1]))


# Urutkan label klaster sesuai dengan frekuensi terjadinya.
sortedLabels = sorted([n for n in range(numClusters)],
    key=lambda x: -np.sum(clustering == x))


# Inisialisasi K-berarti gambar skala abu-abu; mengatur
warna piksel .
kmeansImage = np.zeros(image.shape[:2], dtype=np.uint8)
for i, label in enumerate(sortedLabels):
    kmeansImage[clustering == label] = int(255 /
```

```
(numClusters - 1)) * i


# Menggabungkan gambar aski dan gambar K-Means, dipisahkan
oleh strip abu-abu
concatImage = np.concatenate((orig,
    193 * np.ones((orig.shape[0], int(0.0625 *
orig.shape[1]), 3), dtype=np.uint8),
    cv2.cvtColor(kmeansImage, cv2.COLOR_GRAY2BGR)), axis=1)
cv2.imshow('Citra Original vs Cluster', concatImage)


if args['output_file']:
    # Buat nama file keluaran timestamped dan tulis gambar
ke disk.
    fileExtension = args['output_format']
    filename =
(datetime.datetime.now().strftime("%Y%m%d%H%M%S")
        + colorSpace + '_c' + args['channels'] + 'n' +
str(numClusters) + '.'
        + fileExtension)
    cv2.imwrite(filename, concatImage)
cv2.waitKey(0)
```

Lampiran 3 *source code* klaster warna menggunakan *k-means*

1. Program color_kmeans.py

```
#Setelah koding ini, Menjalankan hasil klaster warna RGB

#C:\Users\Firdaus>cd C:\skripsi program

#C:\skripsi program>python color_kmeans.py --image
kargo/kargo1.jpg --cluster 3


# import packages yang diperlukan

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import argparse

import utils

import cv2



ap = argparse.ArgumentParser()

ap.add_argument("-i", "--image", required = True, help =
"Path to the image")

ap.add_argument("-c", "--clusters", required = True, type =
int,

      help = "# of clusters")

args = vars(ap.parse_args())



image = cv2.imread(args["image"])

image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


# Menunjukkan gambar

plt.figure()

plt.axis("off")

plt.imshow(image)
```

```
image = image.reshape((image.shape[0] * image.shape[1], 3))

# klasterisasi piksel
clt = KMeans(n_clusters = args["clusters"])
clt.fit(image)


hist = utils.centroid_histogram(clt)
bar = utils.plot_colors(hist, clt.cluster_centers_)

# tunjukkan bar warna
plt.figure()
plt.axis("off")
plt.imshow(bar)
plt.show()
```

2. Program util.py

```
import numpy as np
import cv2

def centroid_histogram(clt):
      #ambil jumlah klaster berbeda dan buat histogram
      #berdasarkan jumlah piksel untuk setiap klaster
      numLabels = np.arange(0, len(np.unique(clt.labels_)) +
1)
      (hist, _) = np.histogram(clt.labels_, bins =
numLabels)

      #Normalkan histogram, sehingga jumlahnya menjadi satu
      hist = hist.astype("float")
      hist /= hist.sum()
```

```
    return hist


def plot_colors(hist, centroids):
    # inisialisasi bagan batang yang mewakili frekuensi
relatif masing-masing warna


    bar = np.zeros((50, 300, 3), dtype = "uint8")
    startX = 0


    # mengulangi presentase masing-masing klaster dan
warna masing-masing klaster
    for (percent, color) in zip(hist, centroids):
        # plot presentase setiap klaster
        endX = startX + (percent * 300)
        cv2.rectangle(bar,(int(startX),0),(int(endX),50),
        color.astype("uint8").tolist(), -1)
        startX = endX


    return bar
```