DAFTAR PUSTAKA

Ahmad Abul Khair, Zahir Zainuddin, Andani Achmad, Amil Ahmad Ilham, n.d. Face Recognition in Kindergarten Students using the Principal Component Analysis Algorithm.

Ari Kurniawan, Marzuki Syahfirin, 2016. Aplikasi Deteksi Objek Menggunakan Histogram Of Oriented Gradient Untuk Modul Sistem Cerdas Pada Robot Nao. https://doi.org/10.13140/RG.2.1.3592.9207

Dohyung Kim, Dong-Hyeon Kim, Keun-Chang Kwak, 2017. Classification of K-Pop Dance Movements Based on Skeleton Information Obtained by a Kinect Sensor. www.mdpi.com/journal/sensors. https://doi.org/doi:10.3390/s17061261

Drs. Ahmad susanto, MPd, 2011. PERKEMBAGAN ANAK USIA DINI. Kencana Jkt.

M. A. mursid, 2017. Pengembangan Pembelajaran Paud. PT Remaja Rosdakarya.

Muhammad Furqan Rasyid, Zahir Zainuddin, Andani, 2019. Early Detection of Health Kindergarten Student at School Using Image Processing Technology. ICOST 2019. https://doi.org/DOI 10.4108/eai.2-5-2019.2284609

Navneet Dalal, 2003. Histograms of Oriented Gradients for Human Detection.

Permendikbud137-2014StandarNasionalPAUD.pdf [WWW Document], n.d. URL http://luk.tsipil.ugm.ac.id/atur/bsnp/Permendikbud137-2014StandarNasionalPAUD.pdf (accessed 11.22.19).

Rafael C. Gonzalez,, Richard E. Woods, 2008. Digital Image Processing: Int. Ed. 3rd Ed.

Simone Bianco, Francesco Tisato, 2013. “Karate Moves Recognition from Skeletal Motion,.” Three-Dimens Image Process 3DIP 8650, 86500K,. https://doi.org/10.1117/12.2006229.

Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. 45, 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Thattapon Surasak, Ito Takahiro, Cheng-hsuan Cheng, Chi-en Wang, Pao-you Sheng, n.d. Histogram of Oriented Gradients for Human Detection in Video. 2018 5th Int. Conf. Bus. Ind. Res. ICBIR Bangk. Thail.

Trinh Hoai An, Nguyen Thanh Hai, Truong Quang Phuc, Tran Thanh Mai, n.d. Support Vector Machine algorithm for human fall recognition Kinect-Based skeletal data. 2015 2nd Natl. Found. Sci. Technol. Dev. Conf. Inf. Comput. Sci.

Youness CHOUBIK, Abdelhak MAHMOUDI, n.d. Machine learning for real me poses classification using Kinect skeleton data. 2016 13th Int. onf. Comput. Graph. Imaging Vis. https://doi.org/DOI 0.1109/CGiV.2016.66

ddin, A S Laswi, n.d. Implementation of The LDA Algorithm for nline Validation Based on Face Recognition. Int. Conf. Comput. ppl. Inform. 2016, IOP Conf. Series: Journal of Physics: Conf. eries 801 (2017) 012047. https://doi.org/doi:10.1088/1742-6596/801/1/012047

LISTING PROGRAM

“Index.py”

```
from PIL import ImageTk, Image
import tkinter as tk
from tkinter import font  as tkfont
import testing_program as tp
import conn
# import tkMessageBox
# import tkinter.messagebox
from tkinter import messagebox as msg
from tkinter import ttk

class SampleApp(tk.Tk):

        def __init__(self, *args, **kwargs):
                tk.Tk.__init__(self, *args, **kwargs)

                self.title_font = tkfont.Font(family='Verdana', size=12, weight="bold")
                self.text_font = tkfont.Font(family='Verdana', size=9, weight="bold")

                # the container is where we'll stack a bunch of frames
                # on top of each other, then the one we want visible
                # will be raised above the others
                container = tk.Frame(self)
                self.geometry("960x480")
                self.title("Sistem Klasifikasi Gerakan Anak PAUD")
                self.membuatMenu()
                container.pack(side="top", fill="both", expand=True)
                container.grid_rowconfigure(0, weight=1)
                container.grid_columnconfigure(0, weight=1)

                self.frames = {}
                for F in (StartPage, PageOne, PageTwo):
                        page_name = F.__name__
                        frame = F(parent=container, controller=self)
                        self.frames[page_name] = frame

                        # put all of the pages in the same location;
                        # the one on the top of the stacking order
                        # will be the one that is visible.
                        frame.grid(row=0, column=0, sticky="nsew")

                self.show_frame("StartPage")

        def exportTxt(self):
                f = open(self.headerTitle +'.txt','a')
                f.write(self.fileExport)
                f.close()
                self.info("Info","Data Telah di Export \nDi "+self.headerTitle)
                # self.perintahKeluar()

        rData(self):
                conn.clearData()
                self.info("Info","Data Telah diHapus")

        def info(self, title, content):
```

```
            print('berfungsi',self.headerTitle)
            msg.showinfo(title, content)

    def frameLog(self):
            window = tk.Toplevel(self)
            result = conn.logData()
            header = "LOG DATA HASIL PENGENALAN"
            display = tk.Label(window, text=header, font=self.text_font)
            display.pack(pady=20)
            window.geometry("720x320")
            window.maxsize(720, 500)
            window.title("Log Data Gerakan")
            textAnalyze = tk.Text(window, width=87)
            scrollAnalyze = tk.Scrollbar(window)
            data = []
            data = '\n'.join([str(hasilAnalisis) for hasilAnalisis in result ])
            textAnalyze.insert(tk.END, data)
            print(60*'+')
            print(data)
            print(60*'+')
            self.fileExport = header+'\n\n'+ data
            self.headerTitle = "LOG DATA HASIL PENJUALAN "
            exportBtn = tk.Button(window, text ="EXPORT", command = self.exportTxt)
            exportBtn.place(x = 580, y = 20)

            clearBtn = tk.Button(window, text ="CLEAR", command = self.clearData)
            clearBtn.place(x = 650, y = 20)

            textAnalyze.pack(side=tk.LEFT,fill=tk.Y)
            scrollAnalyze.pack(side=tk.RIGHT,fill=tk.Y)
            scrollAnalyze.config(command=textAnalyze.yview)
            textAnalyze.config(yscrollcommand=scrollAnalyze.set)
            # if (status == 'Buka'):
            #         print('test Buka')
            # elif (status == 'Tutup'):
            #         window.destroy()

    def frameAnalyze(self):
            window = tk.Toplevel(self)
            display = tk.Label(window, text="Klasifikasi Gerakan", font=self.text_font)
            display.pack(pady=20)
            window.geometry("720x320")
            window.maxsize(720, 500)
            window.title("Hasil Klasifikasi Gerakan")
            textAnalyze = tk.Text(window, width=87)
            scrollAnalyze = tk.Scrollbar(window)
            data = []
            result = conn.classificationOfPCA()
            data = '\n'.join([ str(hasilAnalisis) for hasilAnalisis in result ])
            textAnalyze.insert(tk.END, data)
            print(60*'+')
            print(data)
            # print('result =',result)
            print(60*'+')
            textAnalyze.pack(side=tk.LEFT,fill=tk.Y)
            scrollAnalyze.pack(side=tk.RIGHT,fill=tk.Y)
            scrollAnalyze.config(command=textAnalyze.yview)
            textAnalyze.config(yscrollcommand=scrollAnalyze.set)
```

```
        def show_frame(self, page_name):
                '''Show a frame for the given page name'''
                frame = self.frames[page_name]
                frame.tkraise()

        def membuatMenu(self):
                menubar = tk.Menu(self)
                self.config(menu=menubar)

                # fileMenu = tk.Menu(menubar)
                # fileMenu.add_command(label="Cam 0", command=self.callCam(0))
                # menubar.add_cascade(label="Page One", command=lambda:
self.show_frame("PageOne"))
                # menubar.add_cascade(label="Page Two", command=lambda:
self.show_frame("PageTwo"))
                menubar.add_cascade(label="Run Aplikasi", command=self.callCam)
                menubar.add_cascade(label="Hasil Klasifikasi",
command=self.frameAnalyze)
                menubar.add_cascade(label="Log Gerakan Terdeteksi",
command=self.frameLog)

        def callCam(self):
                # self.frameNMove()
                tp.main()

        def perintahKeluar(self):
                self.quit()

class StartPage(tk.Frame):

        def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                self.controller = controller

                tk.Label(self, text="").pack(pady=2)
                tk.Label(self, text="SISTEM KLASIFIKASI PENGENALAN
PERKEMBANGAN GERAKAN PAUD", font=controller.title_font).pack(padx=5, pady=5)

                gambar = Image.open("unhas.png")
                gambar = gambar.resize((200,200), Image.ANTIALIAS)
                gambar = ImageTk.PhotoImage(gambar)
                w = tk.Label(self, image=gambar)
                w.gambar = gambar
                w.pack(pady=25)

                tk.Label(self, text="2020", font=controller.title_font).pack(pady=3,
side="bottom")
                tk.Label(self, text="MAKASSAR", font=controller.title_font).pack(pady=3,

                tk.Label(self, text="UNIVERSITAS HASANUDDIN",
                title_font).pack(pady=3, side="bottom")
                tk.Label(self, text="PROGRAM PASCASARJANA",
                title_font).pack(pady=3, side="bottom")


class PageOne(tk.Frame):
```

```
        def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                self.controller = controller
                label = tk.Label(self, text="This is page 2", font=controller.title_font)
                label.pack(side="top", fill="x", pady=10)
                button = tk.Button(self, text="Go to the start page",
                                                        command=lambda:
controller.show_frame("StartPage"))
                button.pack()


class PageTwo(tk.Frame):

        def __init__(self, parent, controller):
                tk.Frame.__init__(self, parent)
                self.controller = controller
                label = tk.Label(self, text="This is page 2", font=controller.title_font)
                label.pack(side="top", fill="x", pady=10)
                button = tk.Button(self, text="Go to the start page",
                                                        command=lambda:
controller.show_frame("StartPage"))
                button.pack()


if __name__ == "__main__":
        app = SampleApp()
        app.mainloop()
```

“Make_Trained_ Data”

```
import os
import cv2
import numpy as np


def dist(x, y):
    return np.sqrt(np.sum((x - y) ** 2))

def draw_detections(img, rects, thickness = 1):
    for x, y, w, h in rects:
        pad_w, pad_h = int(0.15*w), int(0.05*h)
        cv2.rectangle(img, (x+pad_w, y+pad_h), (x+w-pad_w, y+h-pad_h), (0, 0, 255),
thickness)

if __name__ == '__main__':
    width = 200                 #============> Width Image
    height = 200                #============> Height Image

                                ========================================= My Library
                                OGDescriptor()
                                Detector( cv2.HOGDescriptor_getDefaultPeopleDetector() )

                                ========================================= Input Video
                                ideoCapture('D:/Project Tesis Husna/From WA/1com.avi')
                                eoCapture('data_uji/DATA UJI.MP4')

    nFrame = 0
    contentFile = []
```

```
    while True:
        m = []
        cnt = 0
        nFrame += 1
        print(nFrame,30*'==')

        _, frame = cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        found,w=hog.detectMultiScale(frame, winStride=(8,8), padding=(32,32), scale=1.05)
        draw_detections(frame,found)                #============> Draw Body Detected
        print('found =>', len(found))

        for i in found:
            x,y,w,h = i[0],i[1],i[2],i[3]
            print('x, y, w, h', x, y, w, h)

            gray2 = gray[y:y+h, x:x+w]              #============> Crop Img Person
Founded
            gray2 = cv2.resize(gray2, (200,200))       #============> Resize to 200 x 200

            # ============================================== Save Image
Latih in Folder
            cv2.imwrite('data_latih/found_person/frame_' + str(nFrame) + '_x_'+str(x)+'.png',
gray2)
            # cv2.imshow('Test Image', gray2)

        frame = cv2.resize(frame, (720,640))
        cv2.imshow('student gesture', frame)

        key = cv2.waitKey(1) & 0xff
        if key == ord('q'):
            break


    cap.release()
    cv2.destroyAllWindows()
```

“Training_Program”

```
import os
import cv2
import numpy as np


def unFlatten(vector, rows, cols):
    img = []
    cutter = 0
    while (cutter + cols <= rows * cols):

        img.append(vector[cutter:cutter + cols])

        cutter = vector[cutter:cutter + cols]
        img = cols
    img = np.array(img)


w,h = 200, 200          #============> Resolustion Recomended of Width and Height
```

```
Image

# ================================================ Named Folder from
Data Latih
body = ['NEGATIF','GERAKAN_1',
'GERAKAN_2','GERAKAN_3','GERAKAN_4','GERAKAN_5','GERAKAN_6','GERAKAN_7','
GERAKAN_8']


def main():
    for v in body:
        in_matrix = None
        imgcnt = 0
        print('Read from: ' + v + ' Directory ')
        for f in os.listdir(os.path.join('DATA_LATIH/', v)):
            imgcnt += 1
            print(f)

            # ================================================ Read the Image
in as a gray level image.
            img = cv2.imread(os.path.join('DATA_LATIH/', v, f), cv2.IMREAD_GRAYSCALE)
            img_resized = cv2.resize(img, (w, h))

            # ================================================ Let's resize
them to w * h
            vec = img_resized.reshape(w * h)

            # ================================================ Stack them up
to form the matrix
            try:
                in_matrix = np.vstack((in_matrix, vec))
            except:
                in_matrix = vec

        # ================================================ PCA Method
        if in_matrix is not None:
            mean, eigenvectors = cv2.PCACompute(in_matrix, np.mean(in_matrix,
axis=0).reshape(1, -1))

        img = unFlatten(mean.transpose(), w, h)                #============> Reconstruct
mean to represent an image
        cv2.imwrite('pca_trained/pca_body_' + v + '.png', img)      #============> Save
Trained Image

    cv2.waitKey(0)
    cv2.destroyAllWindows()


#####################################################################
################
= "__main__":
```


ram"

```
import cv2
import numpy as np
```

```
import datetime
import time
import conn

def dist(x, y):
        return np.sqrt(np.sum((x - y) ** 2))

def draw_detections(img, rects, thickness = 1):
        for x, y, w, h in rects:
                # the HOG detector returns slightly larger rectangles than the real objects.
                # so we slightly shrink the rectangles to get a nicer output.
                pad_w, pad_h = int(0.15*w), int(0.05*h)
                cv2.rectangle(img, (x+pad_w, y+pad_h), (x+w-pad_w, y+h-pad_h), (0, 0,
255), thickness)

def findcountur():
        ret, frame1 = cap.read()
        ret, frame2 = cap.read()
                        # masih salah di find countur, bukan video yang sudah di crop yang
di olah
        d = cv2.absdiff(frame1, frame2)
        grey = cv2.cvtColor(d, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(grey, (5, 5), 0)
        ret, th = cv2.threshold( blur, 20, 255, cv2.THRESH_BINARY)
        dilated = cv2.dilate(th, np.ones((3, 3), np.uint8), iterations=1 )
        eroded = cv2.erode(dilated, np.ones((3, 3), np.uint8), iterations=1 )
        c, h = cv2.findContours(eroded, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        cv2.drawContours(frame1, c, -1, (0, 0, 255), 2)
        cv2.putText(frame1, '{+} children status: %s' % 'berkembang', (10,20),
cv2.FONT_HERSHEY_SIMPLEX , 0.5, (0, 0, 255), 2)
                        #show findcountur
        # cv2.imshow("Original", frame2)
        cv2.imshow("Output", frame1)
        cv2.imshow("Output grey", grey)
        cv2.imshow("Output blur", blur)
        cv2.imshow("Output th", th)
        cv2.imshow("Output dilated", dilated)
        cv2.imshow("Output eroded", eroded)

def main():

        width = 200             #============> Width Image
        height = 200    #============> Height Image

        font = cv2.FONT_HERSHEY_SIMPLEX                 #============> Font of Text
        text = ['Negatif','Gerakan 1', 'Gerakan 2','Gerakan 3','Gerakan 4','Gerakan
5','Gerakan 6','Gerakan 7','Gerakan 8']

                ========================================== My Library
                v2.HOGDescriptor()
                SVMDetector( cv2.HOGDescriptor_getDefaultPeopleDetector() )
                cascade = cv2.CascadeClassifier("haarcascade_fullbody.xml")

                ========================================== Input Video
                v2.VideoCapture('D:/Project Tesis Husna/From WA/1com.avi')
        #============> One Person
        cap=cv2.VideoCapture('data_uji/DATA UJI.MP4')
```

```
        #============> Multi Person
        # cap=cv2.VideoCapture('D:/Project Tesis Husna/Program Husna/DATA UJI/DATA
UJI.MP4')

        start_time = datetime.datetime.now().time().strftime('%H:%M:%S')
        #============> Start Time
        nFrame = 0
        contentFile = []
        m = []

        while True:
                cnt = 0
                nFrame += 1
                print('Frame',nFrame,30*'==')


                _, frame = cap.read()
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                found,w=hog.detectMultiScale(frame, winStride=(8,8), padding=(32,32),
scale=1.05)
                #faces = body_cascade.detectMultiScale(gray, scaleFactor = 1.1,
minNeighbors = 5, minSize = (30, 30), flags = cv2.CASCADE_SCALE_IMAGE)

                print('Found =>',len(found))
                # draw_detections(frame,found)              #============> Draw Body
Detected

                for i in range(len(found)):
                        nameTest = 'Anak ke - '+str(i+1)
                        x,y,w,h = found[i][0],found[i][1],found[i][2],found[i][3]

                        img_uji_gray = gray[y:y+h, x:x+w]
                        img_uji_resize = cv2.resize(img_uji_gray, (width,height))
                        img_uji_blurred = cv2.GaussianBlur(img_uji_resize, (5, 5), 0)
                        img_uji_thresh = cv2.adaptiveThreshold(img_uji_blurred, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)

                        bobot_uji = img_uji_thresh.reshape((1, width * height))
                        mean_uji = bobot_uji.mean()

                        vowels = ['NEGATIF','GERAKAN_1',
'GERAKAN_2','GERAKAN_3','GERAKAN_4','GERAKAN_5','GERAKAN_6','GERAKAN_7','
GERAKAN_8']
                        for v in vowels:
                                # print('Read ' + v )
                                f = 'pca_trained/pca_body_' + v + '.png'

                                img_latih_gray = cv2.imread(f, cv2.IMREAD_GRAYSCALE)
                                img_latih_resized = cv2.resize(img_latih_gray, (width,

                                img_latih_blurred = cv2.GaussianBlur(img_latih_resized, (5,

                                img_latih_thresh = cv2.adaptiveThreshold(img_latih_blurred,
TIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
                                bobot_latih = img_latih_thresh.reshape((1, width * height))

                                #
========================================== Distance Vector
```

```
                                distance = dist(bobot_uji, bobot_latih)
                                # print('distance',distance)
                                m.append(distance);
        #============> Euclidean Distance Array
                                # print('m',m)
                                p = (min(m))
        #============> Min Euclidean Distance
                                # print('p',p)
                                pos = m.index(p)
        #============> Array Position
                                # print('pos',pos)
                                result = text[pos]
        #============> Result recognized
                                # cv2.putText(frame, result, (x +- 10, y - 10),
cv2.FONT_HERSHEY_PLAIN, 1, (0, 255, 255))

                        m.clear()                       #============> Remove List
Euclidean Distance Array that more length of vowels

                        # time.sleep(5)
                        now_date = datetime.datetime.now().date().strftime('%Y-%m-%d')
                        end_time = datetime.datetime.now().time().strftime('%H:%M:%S')
                #============> End Time
                        total_time = (datetime.datetime.strptime(end_time,'%H:%M:%S') -
datetime.datetime.strptime(start_time,'%H:%M:%S'))
        #============> Delta Time

                        diff_time = total_time.seconds
        #============> Delta Time in second
                        mod_time = diff_time % 5
        #============> Modulo Time

                        if ((result != 'Negatif') & (mod_time == 0)):
                        # if (mod_time == 0):
                                #
================================================== Begin Write Text in
Frame
                                pad_w, pad_h = int(0.15*w), int(0.05*h)
                                text_offset_x = x+pad_w
                                text_offset_y = y+pad_h - 10
                                (text_width, text_height) = cv2.getTextSize(result,
cv2.FONT_HERSHEY_COMPLEX, fontScale=1, thickness=1)[0]
                                box_coords = ((text_offset_x, text_offset_y), (text_offset_x +
text_width - 2, text_offset_y - text_height - 2))

                                cv2.rectangle(frame, box_coords[0], box_coords[1], (0, 255,
0), cv2.FILLED)
                                cv2.putText(frame, result, (x+pad_w, y+pad_h - 10),
RSHEY_PLAIN, 2, (0, 0, 255))
                                cv2.rectangle(frame, (x+pad_w, y+pad_h), (x+w-pad_w, y+h-
, 0), 2)
                                #
========================================== End Write Text in Frame

                                conn.inputData(now_date, end_time, nameTest, result)
=======> Input to Database
                                cv2.imshow(nameTest, img_uji_thresh)
                #============> Show Frame Thresh Uji
```

```
                                # # hasil = 'Frame '+ str(nFrame) +' || Hasil pengenalan
=>'+result
                                hasil = end_time +' => '+nameTest +' || terdeteksi => '+ result
                                print(hasil)
                                contentFile.append(hasil)
                                cv2.imwrite('Frame_Detected/Frame ' + str(nFrame) +
'_'+str(nameTest)+ '_'+str(result)+'.png', frame)


                        # ============================================== Begin
Write Text in file Txt
                        f = open('hasil.txt','w')
                        f.write('Hasil Klasifikasi Pengenalan Gerakan\n\n')
                        f.write(str(contentFile))
                        f.close()
                        # ============================================== End
Write Text in file Txt

                        # print(str(contentFile))

                        frame = cv2.resize(frame, (720,480))
                        cv2.imshow('Hasil Deteksi Gerakan', frame)

                        key = cv2.waitKey(1) & 0xff
                        if key == ord('q'):
                                break


        cap.release()
        cv2.destroyAllWindows()


##########################################################################################
############################
if __name__ == "__main__":
    main()
```