```python
#Script di unione dei file, prima iterazione analisi video

# -*- coding: utf-8 -*-
"""
Created on Mon Jul  9 14:56:43 2018

@author: Alberto
"""

import argparse
import pandas as pd
import glob

parser = argparse.ArgumentParser(
        description='Inserire il nome della cartella dei file')
parser.add_argument('file_time', help='Nome cartella dei file')
args = parser.parse_args()

sensor_path = args.file_time + "/can" + args.file_time + ".csv"
avm_path = args.file_time + "/avm" + args.file_time + ".csv"
pax_path = args.file_time + "/pax" + args.file_time + ".xlsx"

df_sensor = pd.read_csv(sensor_path, sep = ";", skipinitialspace=True,
                        usecols = ["TIME ", "MESSAGEID ", "VALUE"],
                        converters = {"VALUE": lambda x: int(x, 16)})
df_sensor.columns = df_sensor.columns.str.strip()
df_sensor = df_sensor[(df_sensor["VALUE"] <= 10000) & (df_sensor["VALUE"] >= 1000)]
df_sensor = df_sensor.drop_duplicates().reset_index(drop=True).rename(index=str,
columns={"TIME" : "TIMESTAMP"})

df_avm = pd.read_csv(avm_path, sep = ";", skipinitialspace=True,
                     usecols =["TIMESTAMP ", "DOORS "])
df_avm.columns = df_avm.columns.str.strip()
df_avm = df_avm.drop_duplicates().reset_index(drop=True)

pax_excel = pd.ExcelFile(pax_path)

video_path = "analisi_video/"
video_files = glob.glob(video_path + "*.xlsx")
df_list_video = [pd.DataFrame(columns=['CAM1','TIMESTAMP']),
pd.DataFrame(columns=['CAM2','TIMESTAMP']),
pd.DataFrame(columns=['CAM3','TIMESTAMP'])]
for file_name in video_files:
    file_name_cut = file_name[14:]
    if df_avm.iloc[0,0] <= int(file_name_cut[:10]) <= df_avm.iloc[-1,0]:
        cam = int(file_name_cut[14])
        df_video = pd.ExcelFile(video_path + file_name_cut).parse(sheet_name = 0,
        usecols =("B, E")).astype(int)
        df_video.columns  = ['CAM' + str(cam), 'TIMESTAMP']
        df_list_video[cam-1] = df_list_video[cam-1].append(df_video,
        ignore_index=True)

pos = 1
while pos < len(df_avm.index):
    if ((df_avm.iloc[pos]["TIMESTAMP"]) - (df_avm.iloc[pos-1]["TIMESTAMP"]))>1:
        new_timestamp = df_avm.iloc[pos-1]["TIMESTAMP"] + 1
        new_doors = df_avm.iloc[pos-1]["DOORS"]
        df_avm = df_avm.append(pd.Series([new_timestamp, new_doors],
        index=["TIMESTAMP", "DOORS"]),
        ignore_index=True).sort_values(by=["TIMESTAMP"])
    pos += 1

for df_cam in df_list_video:
    df_cam["TIMESTAMP"] = df_cam["TIMESTAMP"].apply(int)
    df_avm = df_avm.merge(df_cam, on = "TIMESTAMP", how ="left")

open_start = 0
open_end = 0
if (df_avm.iloc[0]["DOORS"] == 0):
   open_start = df_avm.iloc[0]["TIMESTAMP"]
lenght = len(df_avm.index)
df_avm["CHANGE"] = ""
```

```
for x in range (1, lenght):
    if (df_avm.iloc[x]["DOORS"] != df_avm.iloc[x-1]["DOORS"]):
       if (df_avm.iloc[x]["DOORS"] == 0):
           open_start = df_avm.iloc[x]["TIMESTAMP"]
       else:
           df_avm.at[x, "CHANGE"] = 18000
           if (open_start != 0):
               open_end = df_avm.iloc[x]["TIMESTAMP"]
               for x in range (1,4):
                   cam = "CAM" + str(x)
                   mode_calc = df_avm[(df_avm["TIMESTAMP"] >= open_start) &
                   (df_avm["TIMESTAMP"] <= open_end)][cam].mode()
                   if (mode_calc.empty):
                       df_avm[(df_avm["TIMESTAMP"] >= open_start) &
                       (df_avm["TIMESTAMP"] <= open_end)][cam] = 0
                   else:
                       df_avm[(df_avm["TIMESTAMP"] >= open_start) &
                       (df_avm["TIMESTAMP"] <= open_end)][cam] = mode_calc[0]
               open_start = 0
df_avm = df_avm.fillna(method = "ffill")
df_avm = df_avm[(df_avm["DOORS"] == 1)]

df_result = df_avm
for sensor_number in range(1, 5):
    df_sensor_single = df_sensor[df_sensor["MESSAGEID"] ==
    (180+sensor_number)].drop("MESSAGEID",
    1).groupby("TIMESTAMP").max().reset_index().rename(index=str, columns={"VALUE" :
    "SENSOR_" + str(sensor_number)})
    df_result = df_result.merge(df_sensor_single, on = "TIMESTAMP", how ="left")

df_result = df_result.fillna(method="ffill")
col_sum = df_result.loc[:, "SENSOR_1":"SENSOR_4"].sum(axis = 1)
df_result["SUM"] = col_sum
df_result["SUM_FILTER"] = col_sum

lenght = len(df_result.index)
for x in range (3, lenght):
    if (df_avm.iloc[x]["CHANGE"] == 18000):
        df_result.at[x-2, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
        df_result.at[x-1, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]

writer = pd.ExcelWriter(args.file_time + "/results_complete_rides" + args.file_time
+ ".xlsx", engine="xlsxwriter")

for x in range ((len(pax_excel.sheet_names)-1), -1, -1):
    df_pax = pax_excel.parse(sheet_name = x, usecols =("B, I:Q"))
    df_pax = df_pax.drop_duplicates().reset_index(drop=True)
    df_pax.columns = df_pax.columns.str.strip()
    df_result_temp = df_result[(df_result["TIMESTAMP"] >=
    df_pax.iloc[0]["TIMESTAMP"]) & (df_result["TIMESTAMP"] <=
    df_pax.iloc[-1]["TIMESTAMP"])]
    df_result_temp = df_result_temp.merge(df_pax, on = "TIMESTAMP", how ="left")
    df_result_move = df_result_temp[["TIMESTAMP", "CAM1", "CAM2", "CAM3"]]
    df_result_temp.drop(labels=["CAM1", "CAM2", "CAM3"], axis=1,inplace = True)
    df_result_temp = df_result_temp.merge(df_result_move, on = "TIMESTAMP", how
    ="left")
    df_result_temp = df_result_temp.fillna(method="ffill")
    col_video_sum = df_result_temp.loc[:, "CAM1":"CAM3"].sum(axis = 1)
    df_result_temp["VIDEO_SUM"] = col_video_sum

    if (x == 0):
        result_path = args.file_time + "/results_complete" + args.file_time + ".csv"
    else:
        result_path = args.file_time + "/results_ride" + args.file_time + "[ride " +
        str(len(pax_excel.sheet_names)-x) +"].csv"
        df_result_temp.to_excel(writer, sheet_name = "Corsa" +
        str(len(pax_excel.sheet_names)-x), index = False, freeze_panes = (1, 0))

    df_result_temp.to_csv(result_path, sep=";", index=False, float_format='%0.0f')

writer.save()
writer.close()
```

```
126
127
128 #Script di unione dei file, seconda iterazione analisi video
129
130 # -*- coding: utf-8 -*-
131 """
132 Created on Mon Jul  9 14:56:43 2018
133
134 @author: Alberto
135 """
136
137 import argparse
138 import pandas as pd
139 import glob
140
141 parser = argparse.ArgumentParser(
142         description='Inserire il nome della cartella dei file')
143 parser.add_argument('file_time', help='Nome cartella dei file')
144 args = parser.parse_args()
145
146 sensor_path = args.file_time + "/can" + args.file_time + ".csv"
147 avm_path = args.file_time + "/avm" + args.file_time + ".csv"
148 pax_path = args.file_time + "/pax" + args.file_time + ".xlsx"
149
150 df_sensor = pd.read_csv(sensor_path, sep = ";", skipinitialspace=True,
151                         usecols = ["TIME ", "MESSAGEID ", "VALUE"],
152                         converters = {"VALUE": lambda x: int(x, 16)})
153 df_sensor.columns = df_sensor.columns.str.strip()
154 df_sensor = df_sensor[(df_sensor["VALUE"] <= 10000) & (df_sensor["VALUE"] >= 1000)]
155 df_sensor = df_sensor.drop_duplicates().reset_index(drop=True).rename(index=str,
    columns={"TIME" : "TIMESTAMP"})
156
157 df_avm = pd.read_csv(avm_path, sep = ";", skipinitialspace=True,
158                      usecols =["TIMESTAMP ", "DOORS "])
159 df_avm.columns = df_avm.columns.str.strip()
160 df_avm = df_avm.drop_duplicates().reset_index(drop=True)
161
162 pax_excel = pd.ExcelFile(pax_path)
163
164 video_path = "analisi_video/"
165 video_files = glob.glob(video_path + "*.xlsx")
166 df_list_video = [pd.DataFrame(columns=['CAM1','TIMESTAMP']),
    pd.DataFrame(columns=['CAM2','TIMESTAMP']),
    pd.DataFrame(columns=['CAM3','TIMESTAMP'])]
167 for file_name in video_files:
168     file_name_cut = file_name[14:]
169     if df_avm.iloc[0,0] <= int(file_name_cut[:10]) <= df_avm.iloc[-1,0]:
170         cam = int(file_name_cut[14])
171         df_video = pd.ExcelFile(video_path + file_name_cut).parse(sheet_name = 0,
            usecols =("B, E")).astype(int)
172         df_video.columns  = ['CAM' + str(cam), 'TIMESTAMP']
173         df_list_video[cam-1] = df_list_video[cam-1].append(df_video,
            ignore_index=True)
174
175 pos = 1
176 while pos < len(df_avm.index):
177     if ((df_avm.iloc[pos]["TIMESTAMP"]) - (df_avm.iloc[pos-1]["TIMESTAMP"]))>1:
178         new_timestamp = df_avm.iloc[pos-1]["TIMESTAMP"] + 1
179         new_doors = df_avm.iloc[pos-1]["DOORS"]
180         df_avm = df_avm.append(pd.Series([new_timestamp, new_doors],
            index=["TIMESTAMP", "DOORS"]),
            ignore_index=True).sort_values(by=["TIMESTAMP"])
181     pos += 1
182
183 for df_cam in df_list_video:
184     df_cam["TIMESTAMP"] = df_cam["TIMESTAMP"].apply(int)
185     df_avm = df_avm.merge(df_cam, on = "TIMESTAMP", how ="left")
186
187 #open_start = 0
188 #open_end = 0
189 #if (df_avm.iloc[0]["DOORS"] == 0):
190 #    open_start = df_avm.iloc[0]["TIMESTAMP"]
```

```
lenght = len(df_avm.index)
df_avm["CHANGE"] = ""
for x in range (1, lenght):
    if (df_avm.iloc[x]["DOORS"] != df_avm.iloc[x-1]["DOORS"]):
#         if (df_avm.iloc[x]["DOORS"] == 0):
#             open_start = df_avm.iloc[x]["TIMESTAMP"]
#         else:
        if (df_avm.iloc[x]["DOORS"] != 0):
            df_avm.at[x, "CHANGE"] = 18000
#             df_avm.at[x, "CHANGE"] = 18000
#             if (open_start != 0):
#                 open_end = df_avm.iloc[x]["TIMESTAMP"]
#                 for x in range (1,4):
#                     cam = "CAM" + str(x)
#                     mode_calc = df_avm[(df_avm["TIMESTAMP"] >= open_start) &
(df_avm["TIMESTAMP"] <= open_end)][cam].mode()
#                     if (mode_calc.empty):
#                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
(df_avm["TIMESTAMP"] <= open_end)][cam] = 0
#                     else:
#                         df_avm[(df_avm["TIMESTAMP"] >= open_start) &
(df_avm["TIMESTAMP"] <= open_end)][cam] = mode_calc[0]
#                 open_start = 0
df_avm = df_avm.fillna(method = "ffill")
df_avm = df_avm[(df_avm["DOORS"] == 1)]

df_result = df_avm
for sensor_number in range(1, 5):
    df_sensor_single = df_sensor[df_sensor["MESSAGEID"] ==
    (180+sensor_number)].drop("MESSAGEID",
    1).groupby("TIMESTAMP").max().reset_index().rename(index=str, columns={"VALUE" :
    "SENSOR_" + str(sensor_number)})
    df_result = df_result.merge(df_sensor_single, on = "TIMESTAMP", how ="left")

df_result = df_result.fillna(method="ffill")
col_sum = df_result.loc[:, "SENSOR_1":"SENSOR_4"].sum(axis = 1)
df_result["SUM"] = col_sum
df_result["SUM_FILTER"] = col_sum

lenght = len(df_result.index)
for x in range (3, lenght):
    if (df_avm.iloc[x]["CHANGE"] == 18000):
        df_result.at[x-2, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]
        df_result.at[x-1, "SUM_FILTER"] = df_result.iloc[x-3]["SUM"]

writer = pd.ExcelWriter(args.file_time + "/results_complete_rides" + args.file_time
+ ".xlsx", engine="xlsxwriter")

for x in range ((len(pax_excel.sheet_names)-1), -1, -1):
    df_pax = pax_excel.parse(sheet_name = x, usecols =("B, I:Q"))
    df_pax = df_pax.drop_duplicates().reset_index(drop=True)
    df_pax.columns = df_pax.columns.str.strip()
    df_result_temp = df_result[(df_result["TIMESTAMP"] >=
    df_pax.iloc[0]["TIMESTAMP"]) & (df_result["TIMESTAMP"] <=
    df_pax.iloc[-1]["TIMESTAMP"])]
    df_result_temp = df_result_temp.merge(df_pax, on = "TIMESTAMP", how ="left")
    df_result_move = df_result_temp[["TIMESTAMP", "CAM1", "CAM2", "CAM3"]]
    df_result_temp.drop(labels=["CAM1", "CAM2", "CAM3"], axis=1,inplace = True)
    df_result_temp = df_result_temp.merge(df_result_move, on = "TIMESTAMP", how
    ="left")
    df_result_temp = df_result_temp.fillna(method="ffill")
    col_video_sum = df_result_temp.loc[:, "CAM1":"CAM3"].sum(axis = 1)
    df_result_temp["VIDEO_SUM"] = col_video_sum

    if (x == 0):
        result_path = args.file_time + "/results_complete" + args.file_time + ".csv"
    else:
        result_path = args.file_time + "/results_ride" + args.file_time + "[ride " +
        str(len(pax_excel.sheet_names)-x) +"].csv"
        df_result_temp.to_excel(writer, sheet_name = "Corsa" +
        str(len(pax_excel.sheet_names)-x), index = False, freeze_panes = (1, 0))

```

```
251         df_result_temp.to_csv(result_path, sep=";", index=False, float_format='%0.0f')
252 
253     writer.save()
254     writer.close()
```