TV Pmod Mura Correction Checker

This simple program is to summarize SONY TV Pmod Mura Correction Images.

Author

TVQA member `@Zhang Liang`, 20210830

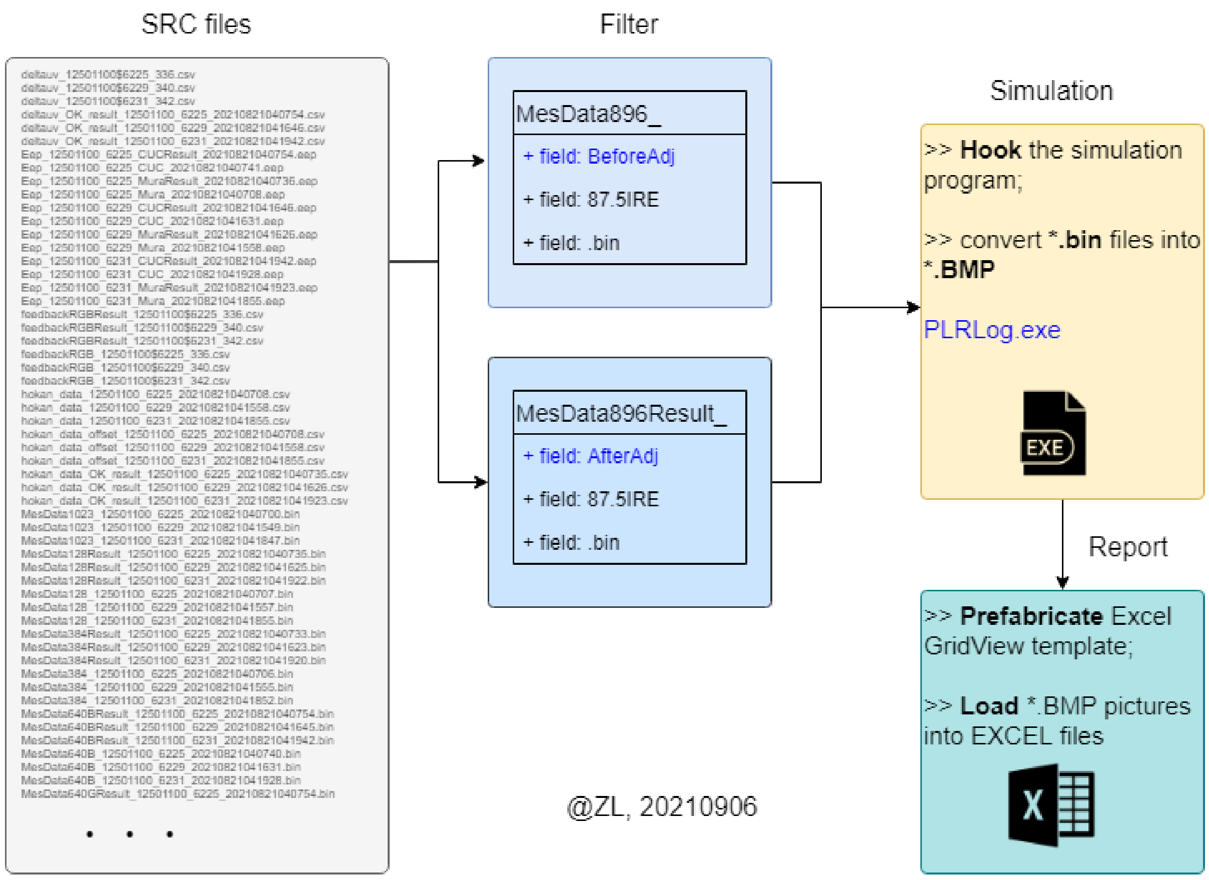
Changelog

- v0.01, initial build
- v0.02, auto-generate report

Final Result

SCC. No	Before Adj, 87.5IRE	After Adj, 87.5IRE
ddddd01	Mura Image	Mura Image
ddddd02	Mura Image	Mura Image
ddddd03	Mura Image	Mura Image
ddddd04	Mura Image	Mura Image
...	...	...
ddddd99	Mura Image	Mura Image

Schema



```
deltauv_12501100$6225_336.csv
deltauv_12501100$6229_340.csv
deltauv_12501100$6231_342.csv
deltauv_OK_result_12501100_6225_20210821040754.csv
deltauv_OK_result_12501100_6229_20210821041646.csv
deltauv_OK_result_12501100_6231_20210821041942.csv
Eep_12501100_6225_CUCResult_20210821040754.eep
Eep_12501100_6225_CUC_20210821040741.eep
Eep_12501100_6225_MuraResult_20210821040736.eep
Eep_12501100_6225_Mura_20210821040708.eep
Eep_12501100_6229_CUCResult_20210821041646.eep
Eep_12501100_6229_CUC_20210821041631.eep
Eep_12501100_6229_MuraResult_20210821041626.eep
Eep_12501100_6229_Mura_20210821041558.eep
Eep_12501100_6231_CUCResult_20210821041942.eep
Eep_12501100_6231_CUC_20210821041928.eep
Eep_12501100_6231_MuraResult_20210821041923.eep
Eep_12501100_6231_Mura_20210821041855.eep
feedbackRGBResult_12501100$6225_336.csv
feedbackRGBResult_12501100$6229_340.csv
feedbackRGBResult_12501100$6231_342.csv
feedbackRGB_12501100$6225_336.csv
feedbackRGB_12501100$6229_340.csv
feedbackRGB_12501100$6231_342.csv
hokan_data_12501100_6225_20210821040708.csv
hokan_data_12501100_6229_20210821041558.csv
hokan_data_12501100_6231_20210821041855.csv
hokan_data_offset_12501100_6225_20210821040708.csv
hokan_data_offset_12501100_6229_20210821041558.csv
hokan_data_offset_12501100_6231_20210821041855.csv
hokan_data_OK_result_12501100_6225_20210821040735.csv
hokan_data_OK_result_12501100_6229_20210821041626.csv
hokan_data_OK_result_12501100_6231_20210821041923.csv
MesData1023_12501100_6225_20210821040700.bin
MesData1023_12501100_6229_20210821041549.bin
MesData1023_12501100_6231_20210821041847.bin
MesData128Result_12501100_6225_20210821040735.bin
MesData128Result_12501100_6229_20210821041625.bin
MesData128Result_12501100_6231_20210821041922.bin
MesData128_12501100_6225_20210821040707.bin
MesData128_12501100_6229_20210821041557.bin
MesData128_12501100_6231_20210821041855.bin
MesData384Result_12501100_6225_20210821040733.bin
MesData384Result_12501100_6229_20210821041623.bin
MesData384Result_12501100_6231_20210821041920.bin
MesData384_12501100_6225_20210821040706.bin
MesData384_12501100_6229_20210821041555.bin
MesData384_12501100_6231_20210821041852.bin
MesData640BResult_12501100_6225_20210821040754.bin
MesData640BResult_12501100_6229_20210821041645.bin
MesData640BResult_12501100_6231_20210821041942.bin
MesData640B_12501100_6225_20210821040740.bin
MesData640B_12501100_6229_20210821041631.bin
MesData640B_12501100_6231_20210821041928.bin
MesData640GResult_12501100_6225_20210821040754.bin
   •   •   •
```

Key points

- (Class) *.bin files Extracter, adds constraints to identify target files;
- (Class) Converter, builds a hook to interact with `PLRLog.exe` to do auto-convert operation, and moves *.BMP images into temporary image folder;
- (Class) MuraImageLoader, loads Mura images into `Excel` by pairs;
- (Function) clear history before new input;

Project Structure

```
D:.
│   20210828 AG65_Pmod_LED_Mura.xlsx
│   20210830 AG65_Pmod_Mura_Image_List.xlsx
│   20210830 AG75_ITC(J)_Mura_Image_List.xlsx
│   20210831 AG75_ITC(J)_Mura_Image_List.xlsx
│   20210901 AG75_ITC(J)_Mura_Image_List.xlsx
│   main.py
│   PLRLog.exe
│   readme.md
│   tmp1.jpg
│   tmp2.jpg
│
├─data
│       MesData896Result_12501100_6225_20210821040730.bin
│       MesData896Result_12501100_6229_20210821041620.bin
│       MesData896Result_12501100_6231_20210821041917.bin
│       MesData896_12501100_6225_20210821040702.bin
│       MesData896_12501100_6229_20210821041551.bin
│       MesData896_12501100_6231_20210821041848.bin
│
├─doc
│       DevManual.md
│       info.txt
│       schema.drawio
│       schema.drawio.png
│
├─Images
│       MesData896Result_12501100_6225_20210821040730.bin.bmp
│       MesData896Result_12501100_6229_20210821041620.bin.bmp
│       MesData896Result_12501100_6231_20210821041917.bin.bmp
│       MesData896_12501100_6225_20210821040702.bin.bmp
│       MesData896_12501100_6229_20210821041551.bin.bmp
│       MesData896_12501100_6231_20210821041848.bin.bmp
│
├─library
│  │  extractor.py
│  │  img.bmp
│  │  plrLog.py
│  │  sn.py
│  │  __init__.py
│  │
│  └─excel
│          image.py
│          __init__.py
│
└─templates
        Mura_List_Template.xlsx
```

Implementation

the following are some core class APIs;

Extractor

```
class Extractor:
    # ids: tuple = ("MesData128_", "MesData384_", "MesData640_", "MesData896_",
"MesData1023_")
    ids:tuple = ("MesData896_", "MesData896Result_")

    def __init__(self, src_folder: str, to_folder: str):
        self.srcFolder = src_folder
        self.toFolder  = to_folder

    def __repr__(self):
        return f"\nSRC folder: {self.srcFolder}\nTo Folder: {self.toFolder}"
    def _indentify(self)->str:
        """
        patter: MesData128_, MesData384_, MesData640_, MesData896_, MesData1023_
        """
        files: list = sorted(pathlib.Path(self.srcFolder).glob("*.bin"))
        for file in files:
            if file.name.startswith(self.ids):
                yield file

    def copy(self):
        for file in self._indentify():
            logging.info(file)
            shutil.copy(file, self.toFolder)
```

ConvertMuraImage

```
class SetLogConverter:
    PLRLOG_OPEN_BTN = (43, 63) # it varies; it depends on PC hardware
    EXT             = '.bmp'
    PATTERN         = '*.bin'
    SLEEP_TIME01    = 2
    SLEEP_TIME_INIT = 6
    SLEEP_TIME02    = 1.5
    SLEEP_TIME03    = 1.5

    def __init__(self, plr_path: str, bin_folder: str, img_path: str, new_folder: 
str):
        self.plr_path   = plr_path
        self.bin_files  = [os.path.abspath(p) for p in 
glob.glob(os.path.join(bin_folder, self.PATTERN))]
        self.img_path   = img_path
        self.new_folder = new_folder

    def to_image(self):
        p = subprocess.Popen(self.plr_path)
        i = 1 # a switch / flag, which makes sure that the first time "open" 
operation wait a long time; otherwise, wait shorter time;
        with temporary_plrlog(p):
            time.sleep(self.SLEEP_TIME01)
            for bin_file in self.bin_files:
                if i > 1:
                    self.load_MesData_bin(bin_file)
                else:
                    self.load_MesData_bin(bin_file, True)
                i += 1

    def rename_img(self, new_name: str)->None:
        os.rename(self.img_path, os.path.abspath(os.path.join(self.new_folder, 
new_name + self.EXT)))
```

```
    def load_MesData_bin(self, bin_file, isInit: bool = False):
        fw = pyautogui.getActiveWindow()
        fw.maximize() # otherwise PLR GUI position changes per opening
        pyautogui.moveTo(*self.PLRLOG_OPEN_BTN) # move mouse to "open"
        pyautogui.click() # click
        if isInit:
            pyautogui.sleep(self.SLEEP_TIME_INIT)
        else:
            pyautogui.sleep(self.SLEEP_TIME02)
        pyautogui.typewrite(bin_file) # input the absolute path of bin file to
address
        pyautogui.sleep(self.SLEEP_TIME03)
        pyautogui.hotkey('Enter') # enter
        pyautogui.sleep(self.SLEEP_TIME03)
        try:
            self.rename_img(os.path.split(bin_file)[-1]) # save img
        except FileExistsError:
            pass
```

MuraImageLoader

```
class MuraImageLoader:
    img_ext      = '.bmp'
    muradata896 = 'mesdata896_'
    muradataresult896 = 'mesdata896result_'
    dstStartRow = 2
    img_quality = 95

    def __init__(self, xl_template:Path, dst_xl:Path, img_folder:Path, width:int,
dstCellName:str, dstCellCol384:str, dstCellCol640:str):
        self._xl_template     = xl_template
        self._dst_xl          = dst_xl
        self._img_folder      = img_folder
        self.width            = width
        self.dstCellName      = dstCellName
        self.dstCellCol384    = dstCellCol384
        self.dstCellCol640    = dstCellCol640
        self._896:List[Path]  = []
        self._896R:List[Path] = []

    def _sort_images(self)->None:
        for root, _, files in os.walk(self._img_folder):
            for file in files:
                if os.path.splitext(file)[-1] == self.img_ext:
                    filepath = os.path.join(root, file)
                    if self.muradata896 in file.lower():
                        self._896.append(filepath)
                    if self.muradataresult896 in file.lower():
```

```
                    self._896R.append(filepath)

    def _is_pmod_id_match(self, x:str, y:str)->bool:
        delimiter = '_'
        idx_id = 2
        return x.split(delimiter)[idx_id] == y.split(delimiter)[idx_id]

    def _export(self)->None:
        idx_w, idx_h = (0, 1)
        i = j = self.dstStartRow
        tmp_img1 = 'tmp1.jpg'
        tmp_img2 = 'tmp2.jpg'
        with open_workbook(self._xl_template, self._dst_xl) as wb:
            ws = wb.worksheets[0]
            if len(self._896) == len(self._896R) and len(self._896) > 0:
                ubound = len(self._896)
                for k in range(0, ubound):
                    img_896 = self._896[k]
                    img_896R = self._896R[k]
                    if not self._is_pmod_id_match(img_896, img_896R):
                        logging.info("PmodIDNotMatchError")
                        break
                    img = Image.open(img_896)
                    width_percent = (self.width/float(img.size[idx_w]))
                    hsize = int((float(img.size[idx_h])*float(width_percent)))
                    img = img.resize((self.width, hsize), Image.ANTIALIAS)
                    img.save(tmp_img1, quality=self.img_quality)
                    img = openpyxl.drawing.image.Image(tmp_img1)
                    dstCellAddress = f'{self.dstCellName}{i}'
                    ws[dstCellAddress].value = os.path.split(img_896)[-1]
                    dstCellAddress = f'{self.dstCellCol384}{i}'
                    ws.add_image(img, dstCellAddress)
                    i += 1

                    img = Image.open(img_896R)
                    width_percent = (self.width/float(img.size[idx_w]))
                    hsize = int((float(img.size[idx_h])*float(width_percent)))
                    img = img.resize((self.width, hsize), Image.ANTIALIAS)
                    img.save(tmp_img2, quality=self.img_quality)
                    img = openpyxl.drawing.image.Image(tmp_img2)
                    dstCellAddress = f'{self.dstCellCol640}{j}'
                    ws.add_image(img, dstCellAddress)
                    j += 1

    def clean(self)->None:
        self._896.clear()
        self._896R.clear()

    def work(self)->None:
        self._sort_images()
        self._export()
```

About