
사진 이미지 분석 도구 제작



Track	Digital Forensic
Category	Tech_03
Nick Name	L3ad0xFF

목 차

1. 개발 목적	1
2. EXIF 정보 추출 하여 EXCEL FILE로 저장.....	1
2.1 사진 내 EXIF 정보 추출	1
2.2 EXCEL FILE 생성	1
2.3 EXCEL FILE을 이용한 그림파일의 EXIF TAG VALUE 분석.....	5
2.4 TAG NAME에 맞는 VALUE 검색과 해당 VALUE를 이용한 GROUPING	8
3. 분석 결과 (2.3의 내용과 비슷).....	11
4. 결론	17
5. 참고문헌	19

1. 개발 목적

- 사진 내 EXIF 정보를 해석한다.
- 다수 사진의 EXIF 정보 중 의미 있는 정보 필터링하여 사진을 그룹핑하는 기능을 구현한다.
- 다수 사진 중 특정 그룹을 선택하여 그룹 사진 파일명 및 추가 Meta 정보 출력을 한다.

2. EXIF 정보 추출 하여 Excel File로 저장

2.1 사진 내 EXIF 정보 추출

```
path = os.getcwd()
for files in os.listdir(path) :
    if (files.split('.')[1] == 'jpg') | (files.split('.')[1] == 'JPG') | (files.split('.')[1] == 'jpeg') | (files.split('.')[1] == 'JPEG') :
        f_list.append(files)

for filename in f_list :
    exif_dict = piexif.load(filename)
    collect_file_tag[filename] = exif_dict

for ifd in ("0th", "Exif", "GPS", "Interop", "1st") :
    for tag in piexif.TAGS[ifd] :
        for filename in f_list :
            for file_tag in collect_file_tag[filename][ifd] :
                if tag == file_tag :
                    exif_list.append(ifd + " : " + piexif.TAGS[ifd][file_tag]["name"] + " : " + filename + " : "
                                     + str(collect_file_tag[filename][ifd][file_tag]))
                    break
```

- 해당 코드가 실행되는 경로에 'jpg, JPG, jpeg, JPEG' 확장자를 검색한 후, 해당 확장자에 일치하는 그림파일이 존재하면 각각 그림파일의 EXIF 값을 exif_list에 [ifd, tag name, filename, tag value] 순으로 분류한다.

2.2 Excel File 생성

```
excel = Workbook()
sheet = excel.active
string = "Tag / Filename"
sheet.title = "0th"
string = "Tag / Filename"
sheet['A1'] = string
sheet['A1'].alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
sheet.freeze_panes = 'A2'

for no_sheet in range(1, 5) :
    if no_sheet == 1 :
        exif_sheet = excel.create_sheet(title = 'Exif')
        exif_sheet['A1'] = string
        exif_sheet['A1'].alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
        exif_sheet.freeze_panes = 'A2'
    elif no_sheet == 2 :
        gps_sheet = excel.create_sheet(title = 'GPS')
        gps_sheet['A1'] = string
        gps_sheet['A1'].alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
        gps_sheet.freeze_panes = 'A2'
    elif no_sheet == 3 :
        interop_sheet = excel.create_sheet(title = 'Interop')
        interop_sheet['A1'] = string
        interop_sheet['A1'].alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
        interop_sheet.freeze_panes = 'A2'
    else :
        _1st_sheet = excel.create_sheet(title = '1st')
        _1st_sheet['A1'] = string
        _1st_sheet['A1'].alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
        _1st_sheet.freeze_panes = 'A2'
```

- 분류한 각 그림파일의 각 EXIF 정보를 Excel에 저장하기 위함이다.
- 1. 현재 활성화 된 sheet의 이름을 ifd 0th로 설정하고, EXIF에서 가장 외부 Key값을 확인 후 존재하는 모든 ifd와 sheet를 1:1 매칭 한다.

```
# => 0th
strlen = list()
for col in range(0, len(f_list)) :
    sheet[chr(chr(col+66)) + "1"] = f_list[col]
    tempcell = sheet[chr(chr(col+66)) + "1"]
    sheet.column_dimensions['A'].width = len(string)
    sheet.column_dimensions[chr(chr(col+66))].width = len(f_list[col])
    tempcell.font = openpyxl.styles.Font(size=10)
    tempcell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempcell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e0cd66'))

for num in range(0, len(_0th_tag_list)) :
    sheet[chr(65) + str(num+2)] = _0th_tag_list[num]
    tempvercell = sheet[chr(65) + str(num+2)]
    tempvercell.font = openpyxl.styles.Font(size=10)
    tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempvercell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e3e3e3'))
    strlen.append(len(_0th_tag_list[num]))
    if num == len(_0th_tag_list)-1 :
        strlen.sort()
        sheet.column_dimensions[chr(65)].width = strlen[num]

strlen = list() #chr(66) = B #split_ exif - [0] : ifd / [1] : tag_name / [2] : filename / [3] : value
for col in range(0, len(f_list)) :
    for num in range(0, len(_0th_tag_list)) :
        for total in range(0, len(_0th_list)) :
            if (sheet[chr(66+col)] + "1".value == _0th_list[total][1]) and (sheet[chr(65) + str(num+2)].value == _0th_list[total][0]) :
                sheet[chr(66+col)] + str(num+2)] = str(_0th_list[total][2])
                tempvercell = sheet[chr(66+col)] + str(num+2)]
                tempvercell.font = openpyxl.styles.Font(size=10)
                tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
```

- Sheet name = 0th인 sheet에 파일에서 추출한 EXIF들 중 IFD = 0th인 tag와 value값을 저장
 1. 각 열 1행에 분석한 모든 파일의 파일 명을 입력한다
 2. 각 행의 A열에 해당 IFD에 존재하는 Tag name을 입력한다.
 3. 파일명과 Tag name에 일치하도록 value를 입력한다.

```
# => Exif
for col in range(0, len(f_list)) :
    exif_sheet[chr(chr(col+66)) + "1"] = f_list[col]
    tempcell = exif_sheet[chr(chr(col+66)) + "1"]
    exif_sheet.column_dimensions['A'].width = len(string)
    exif_sheet.column_dimensions[chr(chr(col+66))].width = len(f_list[col])
    tempcell.font = openpyxl.styles.Font(size=10)
    tempcell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempcell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e0cd66'))

for num in range(0, len(Exif_tag_list)) :
    exif_sheet[chr(65) + str(num+2)] = Exif_tag_list[num]
    tempvercell = exif_sheet[chr(65) + str(num+2)]
    tempvercell.font = openpyxl.styles.Font(size=10)
    tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempvercell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e3e3e3'))
    strlen.append(len(Exif_tag_list[num]))
    if num == len(Exif_tag_list)-1 :
        strlen.sort()
        exif_sheet.column_dimensions[chr(65)].width = strlen[num]

for col in range(0, len(f_list)) :
    for num in range(0, len(Exif_tag_list)) :
        for total in range(0, len(Exif_list)) :
            if (exif_sheet[chr(66+col)] + "1".value == Exif_list[total][1]) and (exif_sheet[chr(65) + str(num+2)].value == Exif_list[total][0]) :
                exif_sheet[chr(66+col)] + str(num+2)] = str(Exif_list[total][2])
                tempvercell = exif_sheet[chr(66+col)] + str(num+2)]
                tempvercell.font = openpyxl.styles.Font(size=10)
                tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
```

- Sheet name = Exif인 sheet에 저장하기 위한 python code 부분이다.

```
# => GPS
strlen = list()
for col in range(0, len(f_list)) :
    gps_sheet[str(chr(col+66)) + "1"] = f_list[col]
    tempcell = gps_sheet[str(chr(col+66)) + "1"]
    gps_sheet.column_dimensions['A'].width = len(string)
    gps_sheet.column_dimensions[str(chr(col+66))].width = len(f_list[col])
    tempcell.font = openpyxl.styles.Font(size=10)
    tempcell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempcell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e0cd66'))

for num in range(0, len(GPS_tag_list)) :
    gps_sheet[str(chr(65)) + str(num+2)] = GPS_tag_list[num]
    tempvercell = gps_sheet[str(chr(65)) + str(num+2)]
    tempvercell.font = openpyxl.styles.Font(size=10)
    tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempvercell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e3e3e3'))
    strlen.append(len(GPS_tag_list[num]))
    if num == len(GPS_tag_list)-1 :
        strlen.sort()
        gps_sheet.column_dimensions[str(chr(65))].width = strlen[num]
```

- Sheet name = GPS인 sheet에 저장하기 위한 python code 부분이다.

1. 위의 코드는 GPS Sheet의 각 열 1행에 파일명 작성과 각 행 A열에 Tag name 작성 부분이며, 타 Sheet 작성 부분과 동일하다.

```
for col in range(0, len(f_list)) :
    for num in range(0, len(GPS_tag_list)) :
        for total in range(0, len(GPS_list)) :
            if (gps_sheet[str(chr(66+col)) + "1"].value == GPS_list[total][1]) and (gps_sheet[str(chr(65)) + str(num+2)].value == GPS_list[total][0]) :
                if GPS_list[total][0] == "GPSLatitude" :
                    latDeg = int(GPS_list[total][2].split(",")[0][2:]) / float(int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]))
                    latMin = int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[total][2].split(",")[3].split(" ")[1][:-1]))
                    latSec = int(GPS_list[total][2].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[total][2].split(",")[5].split(" ")[1][:-2]))
                    Lat = (latDeg + (latMin + latSec / 60.0) / 60.0)
                    GPS_list[total][2] = round(Lat, 6)
                if GPS_list[total][0] == "GPSLongitude" :
                    lonDeg = int(GPS_list[total][2].split(",")[0][2:]) / float(int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]))
                    lonMin = int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[total][2].split(",")[3].split(" ")[1][:-1]))
                    lonSec = int(GPS_list[total][2].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[total][2].split(",")[5].split(" ")[1][:-2]))
                    Lon = (lonDeg + (lonMin + lonSec / 60.0) / 60.0)
                    #GPS_list[total][2] = Lon
                    GPS_list[total][2] = round(Lon, 6)
                if GPS_list[total][0] == "GPSTimeStamp" :
                    clock = round(int(GPS_list[total][2].split(",")[0][2:]) / int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]))
                    minute = round(int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[total][2].split(",")[3].split(" ")[1][:-1])))
                    second = round(int(GPS_list[total][2].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[total][2].split(",")[5].split(" ")[1][:-2])))
                    time = str(clock) + ":" + str(minute) + ":" + str(second)
                    GPS_list[total][2] = time
                if GPS_list[total][0] == "GPSAltitude" :
                    alti = int(GPS_list[total][2].split(",")[0][1:]) / float(int(GPS_list[total][2].split(",")[1].split(" ")[1][:-1]))
                    GPS_list[total][2] = alti
                gps_sheet[str(chr(66+col)) + str(num+2)] = str(GPS_list[total][2])
                tempvercell = gps_sheet[str(chr(66+col)) + str(num+2)]
                tempvercell.font = openpyxl.styles.Font(size=10)
                tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
```

2. GPS 부분의 특징 : GPS의 Latitude(위도), Longitude(경도), Altitude(고도), TimeStamp(시간)의 포맷을 분석에 용이하도록 변경이 되도록 설정하는 부분이다.

아래는 위,경도 변환 시 사용한 수식이다.

- 위도 : $Lat = (latDeg + (latMin + latSec / 60.0) / 60.0)$
- 경도 : $Lon = (lonDeg + (lonMin + lonSec / 60.0) / 60.0)$

```
# => Interop
strlen = list()
for col in range(0, len(f_list)) :
    interop_sheet[str(chr(col+66)) + "1"] = f_list[col]
    tempcell = interop_sheet[str(chr(col+66)) + "1"]
    interop_sheet.column_dimensions['A'].width = len(string)
    interop_sheet.column_dimensions[str(chr(col+66))].width = len(f_list[col])
    tempcell.font = openpyxl.styles.Font(size=10)
    tempcell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempcell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e0cd66'))

for num in range(0, len(Interop_tag_list)) :
    interop_sheet[str(chr(65)) + str(num+2)] = Interop_tag_list[num]
    tempvercell = interop_sheet[str(chr(65)) + str(num+2)]
    tempvercell.font = openpyxl.styles.Font(size=10)
    tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempvercell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e3e3e3'))
    strlen.append(len(Interop_tag_list[num]))
    if num == len(Interop_tag_list)-1 :
        strlen.sort()
        interop_sheet.column_dimensions[str(chr(65))].width = strlen[num]

for col in range(0, len(f_list)) :
    for num in range(0, len(Interop_tag_list)) :
        for total in range(0, len(Interop_list)) :
            if (interop_sheet[str(chr(66+col)) + "1"].value == Interop_list[total][1] and (interop_sheet[str(chr(65)) + str(num+2)].value == Interop_list[total][0]) :
                interop_sheet[str(chr(66+col)) + str(num+2)] = str(Interop_list[total][2])
                tempvercell = interop_sheet[str(chr(66+col)) + str(num+2)]
                tempvercell.font = openpyxl.styles.Font(size=10)
                tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
```

- Sheet name = Interop인 sheet에 저장하기 위한 python code 부분이다.

```
# => 1st
strlen = list()
for col in range(0, len(f_list)) :
    _1st_sheet[str(chr(col+66)) + "1"] = f_list[col]
    tempcell = _1st_sheet[str(chr(col+66)) + "1"]
    _1st_sheet.column_dimensions['A'].width = len(string)
    _1st_sheet.column_dimensions[str(chr(col+66))].width = len(f_list[col])
    tempcell.font = openpyxl.styles.Font(size=10)
    tempcell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempcell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e0cd66'))

for num in range(0, len(_1st_tag_list)) :
    _1st_sheet[str(chr(65)) + str(num+2)] = _1st_tag_list[num]
    tempvercell = _1st_sheet[str(chr(65)) + str(num+2)]
    tempvercell.font = openpyxl.styles.Font(size=10)
    tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
    tempvercell.fill = openpyxl.styles.PatternFill(patternType='solid', fgColor=openpyxl.styles.Color('e3e3e3'))
    strlen.append(len(_1st_tag_list[num]))
    if num == len(_1st_tag_list)-1 :
        strlen.sort()
        _1st_sheet.column_dimensions[str(chr(65))].width = strlen[num]

for col in range(0, len(f_list)) :
    for num in range(0, len(_1st_tag_list)) :
        for total in range(0, len(_1st_list)) :
            if (_1st_sheet[str(chr(66+col)) + "1"].value == _1st_list[total][1] and (_1st_sheet[str(chr(65)) + str(num+2)].value == _1st_list[total][0]) :
                _1st_sheet[str(chr(66+col)) + str(num+2)] = str(_1st_list[total][2])
                tempvercell = _1st_sheet[str(chr(66+col)) + str(num+2)]
                tempvercell.font = openpyxl.styles.Font(size=10)
                tempvercell.alignment = openpyxl.styles.Alignment(horizontal = 'center', vertical = 'center')
```

- Sheet name = Interop인 sheet에 저장하기 위한 python code 부분이다.

```
savename = "excel_parser_exif.xlsx"
excel.save(str(path) + "\\ " + str(savename))
print ("Finish the create Excel file for jpeg exif\n")
print ("save path : " + str(path) + "\\ " + str(savename))
```

- 위에서 각 Excel의 행과 열에 입력될 내용을 지정한 Excel을 파일명 : "excel_parser_exif.xlsx"로 설정하고, 해당 python code를 실행한 위치에 저장하도록 한다.

2.3 Excel File을 이용한 그림파일의 Exif Tag value 분석

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
2	Orientation	1	1	1	1	6	6	1	6	1	1	1	1
3	GPSTag	10162	5940	5940	5940	5940	5940	10162	5940	10162	5940	5940	5940
4	DateTime	b'2018:01:09 10:54:25'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2018:01:09 11:16:52'	b'2017:05:23 13:26:11'	b'2018:01:09 11:16:25'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'
5	XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
6	Software	b'Microsoft Windows Photo Viewer 6.1.7600.16385'						b'Microsoft Windows Photo Viewer 6.1.7600.16385'		b'Microsoft Windows Photo Viewer 6.1.7600.16385'			
7	YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
8	Make	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'
9	Model	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'
10	ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
11	YCbCrPositioning	1	1	1	1	1	1	1	1	1	1	1	1
12	ExifTag	2326	184	195	184	195	195	2326	195	2326	195	184	195
13													

- 파일에 존재하는 IFD의 값으로 Excel에 Sheet가 생성됨을 확인 할 수 있다. 또한, A행 2열부터 각 행에는 Tag name이 입력 되어 있으며, B열부터 각 행 1열에는 파일명이 존재한다.

2.3.1 0th

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
Orientation	1	1	1	1	6	6	1	6	1	1	1	1
GPSTag	10162	5940	5940	5940	5940	5940	10162	5940	10162	5940	5940	5940
DateTime	b'2018:01:09 10:54:25'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2018:01:09 11:16:52'	b'2017:05:23 13:26:11'	b'2018:01:09 11:16:25'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'
XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Software	b'Microsoft Windows Photo Viewer 6.1.7600.16385'						b'Microsoft Windows Photo Viewer 6.1.7600.16385'		b'Microsoft Windows Photo Viewer 6.1.7600.16385'			
YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Make	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'
Model	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'
ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
YCbCrPositioning	1	1	1	1	1	1	1	1	1	1	1	1
ExifTag	2326	184	195	184	195	195	2326	195	2326	195	184	195

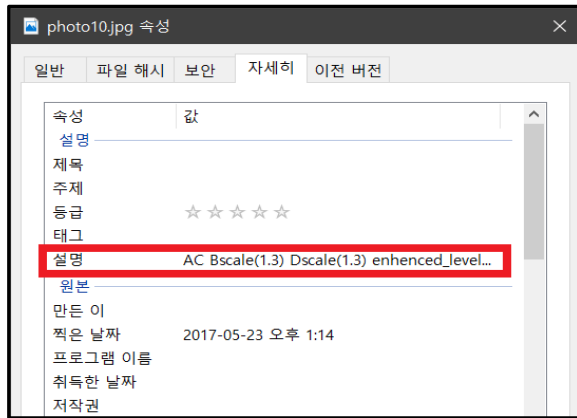
- 각 Tag name에 존재하는 value들 중 대체적인 값은 빨강, 상대적으로 적은 수의 일치 값들을 파랑으로 설정하면 위와 같은 Insight를 얻을 수 있다.
- GPSTag의 부분과 ExifTag 부분의 차이 발생은 해당 Tag들 사이에 존재하는 값들이 다른 그림파일과 길이가 다르며, 사용자 임의 조작으로 인해 길이가 달라 질 수도 있다고 판단.
- 특정 파일만 software가 명시 되며, 다른 파일의 경우 의도적으로 삭제를 했을 수도 있다고 판단
- DateTime이 다른 3가지는 사용자가 시간을 조작 했을 수도 있다고 판단.

2.3.2 Exif

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
SubSecTimeDigitized	b'908354'	b'671598'	b'348982'	b'348982'	b'908354'	b'105920'	b'105920'	b'959989'	b'959989'	b'613968'	b'613968'	b'671598'
FNumber	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)
SceneCaptureType	0	0	0	0	0	0	0	0	0	0	0	0
SubSecTime	b'908354'	b'671598'	b'348982'	b'348982'	b'908354'	b'105920'	b'105920'	b'959989'	b'959989'	b'613968'	b'613968'	b'671598'
ExposureProgram	0	0	0	0	0	0	0	0	0	0	0	0
PixelDimension	2080	272633920	4160	272633920	4160	4160	2080	4160	2080	4160	272633920	4160
ColorSpace	1	1	1	1	1	1	1	1	1	1	1	1
FlashpiVersion	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'
ComponentsConfiguration	b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v; b'w'x01#w'x02#w'x03#v;											
	00'	00'	00'	00'	00'	00'	00'	00'	00'	00'	00'	00'
	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)
	DigitalZoomRatio	2700	50	50	50	2700	2700	2700	2700	50	50	50
	ISOSpeedRatings	2700	50	50	50	2700	2700	2700	2700	50	50	50
ExposureMode	0	0	0	0	0	0	0	0	0	0	0	0
SceneType	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'	b'w'x01'
SensingMethod	2	2	2	2	2	2	2	2	2	2	2	2
BrightnessValue	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)	(0, 100)
PixelDimension	4160	136316960	2080	136316960	2080	2080	4160	2080	4160	2080	136316960	2080
Flash	16	16	16	16	16	16	16	16	16	16	16	16
InteroperabilityTag	10120	5910	5910	5910	5910	5910	10120	5910	10120	5910	5910	5910
ExposureBiasValue	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)	(0, 6)
FocalLength	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)	(403, 100)
DateTimeDigitized	b'2017:05:23 13:26:51'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2017:05:23 13:26:05'	b'2017:05:23 13:26:11'	b'2017:05:23 13:26:11'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'
WhiteBalance	0	0	0	0	0	0	0	0	0	0	0	0
DateTimeOriginal	b'2017:05:23 13:26:51'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2017:05:23 13:26:05'	b'2017:05:23 13:26:11'	b'2017:05:23 13:26:11'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'

Tag / Filename	photo1.jpg	photo2.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
MeteringMode	2	2	2	2	2	2	2	2	2	2	2	2
UserComment	b' A1SHUTTER G547 IN3 N3 x00 Y40 P60.00 C60.00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' AS/CIN/200w0W x00 AC Bscale(1.3) Dscale(1.3) enhanced_Level(7) isOutdoor(0) result(0) FMO	b' AC Bscale(1.3) Dscale(1.3) enhanced_Level(13) isOutdoor(0) result(0) FMO	b' AS/CIN/200w0W x00 AC Bscale(1.3) Dscale(1.3) enhanced_Level(13) isOutdoor(0) result(0) FMO	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' A1SHUTTER G547 IN3 N3 Y00 Y71 CT1 x0 y8Y00 S0 C100 FMO CR1.04	b' HB = 0.0 B/CIN/200w0W x00 HB = 0.0 Dscale(1.3) enhanced_Level(7) isOutdoor(0) = ISO = exp=0.000000 return = FC001000110: b'falc	b' HB = 0.0 B/CIN/200w0W x00 HB = 0.0 Dscale(1.3) enhanced_Level(7) isOutdoor(0) = ISO = exp=0.000000 return = FC001000110: b'falc
	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC001000110: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	FC11111111: b'falc 0000xxxxxxG600K1 0000001212532 8904395R3184502	1635134821 AC Bscale(1.3) Dscale(1.3) enhanced_Level(15) isOutdoor(0) result(0) FMO CR1.04	1635134821 AC Bscale(1.3) Dscale(1.3) enhanced_Level(15) isOutdoor(0) result(0) FMO CR1.04
	ee040194 c2b23 0 0 0 0 0 0 0	01655080x438b10 03461761301040 c2b23 c2b3 0 0 0 0 0 0	01040 924 c2b23 03461761301040 c2b23 c2b3 0 0 0 0 0 0	01040 924 c2b23 03461761301040 c2b23 c2b3 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	ee040194 c2b23 0 0 0 0 0 0 0	03461761301040 c2b23 c2b3 0 0 0 0 0 0	03461761301040 c2b23 c2b3 0 0 0 0 0 0
	0242116b11b 13 12261911194 13	03451151b 0 c 025a519d19 4 b 0 0 0 0 0 0 0	025a519d19 4 b 0 0 0 0 0 0 0	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	0242116b11b 13 12261911194 13	FC000111011: b'falc CR1.04	FC000111011: b'falc CR1.04
	020a117117c 12 021a117115a 22	01ad6795d122 2 11967818d114 2	02785066d127 4 12875066d125 4	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	020a117117c 12 021a117115a 22	0000xxxxxxG600K1 11967818d114 2	0000xxxxxxG600K1 11967818d114 2
	022b11061102 2 024211061102 2	01b8777d12e 2 1187777d12e 2	028b1061025 4 12875066d125 4	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	022b11061102 2 024211061102 2	0000xxxxxxG600K1 11967818d114 2	0000xxxxxxG600K1 11967818d114 2
	0259100e100d 21 0278100a100a 21	1195713ad0f5 2 1196713ad112 2	0265f0dc0d158 4 126650dc0d169 4	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	0259100e100d 21 0278100a100a 21	01767123f15 4 0 0 0 0 0 0 0	01767123f15 4 0 0 0 0 0 0 0
	0287100a100a 21 029e10091009 21	11a7714d112 2 11b0715d2127 2	025a519d19 4 b 125450d1b197 4	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	0287100a100a 21 029e10091009 21	016c708f21 4 01677123f15 4	016c708f21 4 01677123f15 4
	02a510091009 21 02c2c10091009 21	11b0715d2127 2 11a7714d1112 2	125450d1b197 4 125450d1b197 4	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	02a510091009 21 02c2c10091009 21	015d70b0f7 4 015d70b15 4	015d70b0f7 4 015d70b15 4
	02e310091009 21 02e310091009 21	11c27147d106 2 11b7136d08 2	124250d1b197 4 124250d1b197 4	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	0157d0b0f9ac 4 0157d0b0f9ac 4	0157d0b0f9ac 4 0157d0b0f9ac 4
SubSecTimeOriginal ShutterSpeedValue ExposureTime	8908354 (2234, 1000) (21, 100)	b'671598 (5321, 1000) (1, 40)	b'949892 (5321, 1000) (1, 40)	b'949892 (5321, 1000) (1, 40)	8908354 (2234, 1000) (21, 100)	b'105920 (2344, 1000) (1, 40)	b'105920 (2234, 1000) (21, 100)	b'959989 (2344, 1000) (21, 100)	b'959989 (2234, 1000) (21, 100)	b'613968 (5321, 1000) (1, 40)	b'613968 (5321, 1000) (1, 40)	

- X, Y의 크기를 나타내는 부분에서 의심스러운 단위의 수가 입력되어 있으며, 기본적으로 (4160, 2080)이라 판단이 된다. X, Y 쌍이 변경 된 부분은 사진 촬영 시, 가로촬영 또는 세로 촬영의 옵션 부분이라 생각 된다.
- DateTimeOriginal의 경우 사진이 찍힌 본래의 시간이라 판단되며, 앞선 IFD가 0th인 시간 부분과 비교를 하여 위,변조 여부를 판단이 가능하다.
- UserComment 부분에 3개의 파일만 ASCII라고 명시가 되어 있다. 해당 부분 파일의 속성을 확인 하였을 때, 설명 부분에 입력 값이 명시 된다.



2.3.3 GPS

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
GPSLatitudeRef	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'
GPSLatitude	37.027794	37.026524	37.026449	37.026449	37.027794	37.027768	37.027768	37.02777	37.02777	37.026591	37.026591	37.026524
GPSAltitudeRef	0	0	0	0	0	0	0	0	0	0	0	0
GPSTimeStamp	4:1:49		4:1:49		4:1:49	4:1:2	4:1:2	4:1:10	4:1:10	4:1:31		4:1:41
GPSLongitude	127.024582	127.024653	127.024601	127.024601	127.024582	127.024596	127.024596	127.024594	127.024594	127.024498	127.024498	127.024653
GPSDateStamp	b'2017:05:23'		b'2017:05:23'		b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'		b'2017:05:23'
GPSLongitudeRef	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'
GPSAltitude	207.0	0.0	0.0	0.0	207.0	206.0	206.0	206.0	206.0	0.0	0.0	0.0

- 해당 파랑 부분을 보면 3개의 파일은 GPS가 기록 될 때의 날짜와 시간이 명시 되어 있지 않다. 애초 GPS 값이 없는 파일에 의도적으로 GPS 위,경도를 삽입 또는 사진 촬영 시, 기록된 GPS의 시간 값을 삭제 했을 수도 있다고 판단.
- 녹색의 경우 추가적으로 고려 할 수 있는 부분이라 판단하여 색상을 다르게 표시함. 실제로 촬영 시 기기에서 고도를 인식 못하여 기록이 안되었을 수도 있지만, 다른 파일과 비교했을 때, 실제 값이 있는 파일도 존재하는 것으로 보아 0.0으로 기록된 값은 위,변조가 되었을 수도 있다는 가능성이 존재. (상대적으로 중요하게 고려할 단계의 Tag라 판단은 하지 않음)

2.3.4 Interop

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
InteroperabilityIndex	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'

- 의견 없음.

2.3.5 1st

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
Orientation	6	1	1	1	6	6	6	6	6	1	1	1
XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Compression	6	6	6	6	6	6	6	6	6	6	6	6
ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
JPEGInterchangeFormatLength	3341	4044	16441	5109	2819	2801	3334	3907	4412	15167	4533	13532
JPEGInterchangeFormat	10464	6180	6239	6180	6239	6239	10464	6239	10464	6239	6180	6239

- Thumbnail과 관련 있는 tag로 판단되며, 의견없음

2.4 Tag name에 맞는 value 검색과 해당 value를 이용한 Grouping

```

if len(sys.argv) != 3 :
    print ("Usage : value_group [idf key value] [tag name]")
    print ("idf key value : 0th, Exif, GPS, Interop, 1st")

else :
    path = os.getcwd()
    for files in os.listdir(path) :
        if (files.split('.')[1] == 'jpg') | (files.split('.')[1] == 'JPG') | (files.split('.')[1] == 'jpeg') | (files.split('.')[1] == 'JPEG') :
            f_list.append(files)

    for filename in f_list :
        exif_dict = piexif.load(filename)
        collect_file_tag[filename] = exif_dict

    #print (exif_dict.keys())

    for ifd in ("0th", "Exif", "GPS", "Interop", "1st") :
        for tag in piexif.TAGS[ifd] :
            for filename in f_list :
                for file_tag in collect_file_tag[filename][ifd] :
                    if tag == file_tag :
                        exif_list.append(ifd + " : " + piexif.TAGS[ifd][file_tag]["name"] + " : " + str(collect_file_tag[filename][ifd][file_tag]) + " : " + filename)
                        break

```

- [실행할 python code] [IFD] [Tag name]의 양식을 지정하며, 맞지 않을 경우, 오류 메시지를 출력하여 올바른 사용법을 제시한다.
 - Python code가 실행되는 위치에 있는 확장자가 'jpg JPG, jpeg, JPEG'인 모든 파일의 EXIF를 조사한다.
- (Excel file을 읽어서 sheet 별로 정리를 해도 되지만, Excel 파일에 위,변조가 있었을 경우 비교가 어려워져서, 조사 대상파일의 EXIF를 직접 획득 - 무결성)

```

if name1 == "0th" :
    for order in range(0, len(_0th_list)) :
        if name2 in _0th_tag_list :
            if name2 == _0th_list[order][0] :
                compare_value.append(_0th_list[order][1])
                compare_value = list(set(compare_value))
                #print (compare_value)
                if _0th_list[order][1] in compare_value :
                    if len(compare_value) == 1 :
                        fin_val_1.append(_0th_list[order][2])
                        dic_tag_value[compare_value[0]] = fin_val_1
                    elif len(compare_value) == 2 :
                        if compare_value[0] == _0th_list[order][1] :
                            fin_val_1.append(_0th_list[order][2])
                            dic_tag_value[compare_value[0]] = fin_val_1
                        else :
                            fin_val_2.append(_0th_list[order][2])
                            dic_tag_value[compare_value[1]] = fin_val_2
                    elif len(compare_value) == 3 :
                        if compare_value[0] == _0th_list[order][1] :
                            fin_val_1.append(_0th_list[order][2])
                            dic_tag_value[compare_value[0]] = fin_val_1
                        elif compare_value[1] == _0th_list[order][1] :
                            fin_val_2.append(_0th_list[order][2])
                            dic_tag_value[compare_value[1]] = fin_val_2
                        else :
                            fin_val_3.append(_0th_list[order][2])
                            dic_tag_value[compare_value[2]] = fin_val_3
                    elif len(compare_value) == 4 :
                        if compare_value[0] == _0th_list[order][1] :
                            fin_val_1.append(_0th_list[order][2])
                            dic_tag_value[compare_value[0]] = fin_val_1
                        elif compare_value[1] == _0th_list[order][1] :
                            fin_val_2.append(_0th_list[order][2])
                            dic_tag_value[compare_value[1]] = fin_val_2
                        elif compare_value[2] == _0th_list[order][1] :
                            fin_val_3.append(_0th_list[order][2])
                            dic_tag_value[compare_value[2]] = fin_val_3
                        else :
                            fin_val_4.append(_0th_list[order][2])
                            dic_tag_value[compare_value[3]] = fin_val_4
                    elif len(compare_value) == 12 :
                        if compare_value[0] == _0th_list[order][1] :
                            fin_val_1.append(_0th_list[order][2])
                            dic_tag_value[compare_value[0]] = fin_val_1
                        elif compare_value[1] == _0th_list[order][1] :
                            fin_val_2.append(_0th_list[order][2])
                            dic_tag_value[compare_value[1]] = fin_val_2
                        elif compare_value[2] == _0th_list[order][1] :
                            fin_val_3.append(_0th_list[order][2])
                            dic_tag_value[compare_value[2]] = fin_val_3
                        elif compare_value[3] == _0th_list[order][1] :
                            fin_val_4.append(_0th_list[order][2])
                            dic_tag_value[compare_value[3]] = fin_val_4
                        elif compare_value[4] == _0th_list[order][1] :
                            fin_val_5.append(_0th_list[order][2])
                            dic_tag_value[compare_value[4]] = fin_val_5
                        elif compare_value[5] == _0th_list[order][1] :
                            fin_val_6.append(_0th_list[order][2])
                            dic_tag_value[compare_value[5]] = fin_val_6
                        elif compare_value[6] == _0th_list[order][1] :
                            fin_val_7.append(_0th_list[order][2])
                            dic_tag_value[compare_value[6]] = fin_val_7
                        elif compare_value[7] == _0th_list[order][1] :
                            fin_val_8.append(_0th_list[order][2])
                            dic_tag_value[compare_value[7]] = fin_val_8
                        elif compare_value[8] == _0th_list[order][1] :
                            fin_val_9.append(_0th_list[order][2])
                            dic_tag_value[compare_value[8]] = fin_val_9
                        elif compare_value[9] == _0th_list[order][1] :
                            fin_val_10.append(_0th_list[order][2])
                            dic_tag_value[compare_value[9]] = fin_val_10
                        elif compare_value[10] == _0th_list[order][1] :
                            fin_val_11.append(_0th_list[order][2])
                            dic_tag_value[compare_value[10]] = fin_val_11
                        else :
                            fin_val_12.append(_0th_list[order][2])
                            dic_tag_value[compare_value[11]] = fin_val_12
                #print (str(exif_list[order][3]) + " : " + str(exif_l_1_
                else :
                    print ("\nPlease Cheak the tag name!")

```

- 입력한 IFD = 0th일 경우, 해당 IFD에 관한 각 분석 파일의 EXIF로만 검색을 실시하고, 사용자가 입력한 Tag name이 존재하면, Tag name에 맞는 Tag value 값들을 모두 추출한다.
- 추출한 Tag value값들 중 중복되는 것을 제거한 총 개수로 grouping 할 group의 개수를 선정하고, 분석 파일의 Tag value가 일치하는 것끼리 분류한다.

- Tag value를 python dictionary의 key값으로, 해당되는 파일명을 value값으로 dictionary를 생성한다. (이 때, 생성되는 dictionary를 dic_tag_value라고 명명한다.)
- IFD에 해당하는 Tag name이 존재하지 않을 시, "Please Cheak the tag_name!"의 오류메시지를 출력하도록 한다.

```

600 elif name1 == "Exif":
601     compare_value = list()
602     for order in range(0, len(Exif_list)):
603         if name2 in Exif_tag_list:
604             if name2 == Exif_list[order][0]:
605                 compare_value.append(Exif_list[order][1])
606                 compare_value = list(set(compare_value))
607             else:
608                 #print ("Please Check the tag_name!")
609                 #print (compare_value)
610         for order in range(0, len(Exif_list)):
611             if Exif_list[order][1] in compare_value:
612                 #print ("test")
613                 if len(compare_value) == 1:
614                     fin_val_1.append(Exif_list[order][2])
615                     dic_tag_value[compare_value[0]] = fin_val_1
616                 elif len(compare_value) == 2:
617                     if compare_value[0] == Exif_list[order][1]:
618                         fin_val_1.append(Exif_list[order][2])
619                         dic_tag_value[compare_value[0]] = fin_val_1
620                     else:
621                         fin_val_2.append(Exif_list[order][2])
622                         dic_tag_value[compare_value[1]] = fin_val_2
623                 elif len(compare_value) == 3:
624                     if compare_value[0] == Exif_list[order][1]:
625                         fin_val_1.append(Exif_list[order][2])
626                         dic_tag_value[compare_value[0]] = fin_val_1
627                     elif compare_value[1] == Exif_list[order][1]:
628                         fin_val_2.append(Exif_list[order][2])
629                         dic_tag_value[compare_value[1]] = fin_val_2
630                     else:
631                         fin_val_3.append(Exif_list[order][2])
632                         dic_tag_value[compare_value[2]] = fin_val_3
633                 elif len(compare_value) == 4:
634                     #print("4ea")
635                     if compare_value[0] == Exif_list[order][1]:
636                         fin_val_1.append(Exif_list[order][2])
637                         dic_tag_value[compare_value[0]] = fin_val_1
638                     elif compare_value[1] == Exif_list[order][1]:
639                         fin_val_2.append(Exif_list[order][2])
640                         dic_tag_value[compare_value[1]] = fin_val_2
641                     elif compare_value[2] == Exif_list[order][1]:
642                         fin_val_3.append(Exif_list[order][2])
643                         dic_tag_value[compare_value[2]] = fin_val_3
644                     elif compare_value[3] == Exif_list[order][1]:
645                         fin_val_4.append(Exif_list[order][2])
646                         dic_tag_value[compare_value[3]] = fin_val_4
647                     elif compare_value[4] == Exif_list[order][1]:
648                         fin_val_5.append(Exif_list[order][2])
649                         dic_tag_value[compare_value[4]] = fin_val_5
650                     elif compare_value[5] == Exif_list[order][1]:
651                         fin_val_6.append(Exif_list[order][2])
652                         dic_tag_value[compare_value[5]] = fin_val_6
653                     elif compare_value[6] == Exif_list[order][1]:
654                         fin_val_7.append(Exif_list[order][2])
655                         dic_tag_value[compare_value[6]] = fin_val_7
656                     elif compare_value[7] == Exif_list[order][1]:
657                         fin_val_8.append(Exif_list[order][2])
658                         dic_tag_value[compare_value[7]] = fin_val_8
659                     elif compare_value[8] == Exif_list[order][1]:
660                         fin_val_9.append(Exif_list[order][2])
661                         dic_tag_value[compare_value[8]] = fin_val_9
662                     elif compare_value[9] == Exif_list[order][1]:
663                         fin_val_10.append(Exif_list[order][2])
664                         dic_tag_value[compare_value[9]] = fin_val_10
665                     elif compare_value[10] == Exif_list[order][1]:
666                         fin_val_11.append(Exif_list[order][2])
667                         dic_tag_value[compare_value[10]] = fin_val_11
668                     else:
669                         fin_val_12.append(Exif_list[order][2])
670                         dic_tag_value[compare_value[11]] = fin_val_12

```

- IFD가 Exif인 조건일 때이며, 역할은 0th일 경우와 일치한다.

```

elif name1 == "GPS" :
    compare_value = list()
    for order in range(0, len(GPS_list)) :
        if name2 in GPS_tag_list :
            if name2 == GPS_list[order][0] :
                if name2 == "GPSLatitude" :
                    latDeg = int(GPS_list[order][1].split(",")[0][2:]) / float(int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]))
                    latMin = int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[order][1].split(",")[3].split(" ")[1][:-1]))
                    latSec = int(GPS_list[order][1].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[order][1].split(",")[5].split(" ")[1][:-2]))
                    Lat = (latDeg + (latMin + latSec / 60.0) / 60.0)
                    GPS_list[order][1] = round(Lat, 6)
                if name2 == "GPSLongitude" :
                    lonDeg = int(GPS_list[order][1].split(",")[0][2:]) / float(int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]))
                    lonMin = int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[order][1].split(",")[3].split(" ")[1][:-1]))
                    lonSec = int(GPS_list[order][1].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[order][1].split(",")[5].split(" ")[1][:-2]))
                    Lon = (lonDeg + (lonMin + lonSec / 60.0) / 60.0)
                    GPS_list[order][1] = round(Lon, 6)
                if name2 == "GPSTimeStamp" :
                    clock = round(int(GPS_list[order][1].split(",")[0][2:]) / int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]))
                    minute = round(int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]) / float(int(GPS_list[order][1].split(",")[3].split(" ")[1][:-1]))
                    second = round(int(GPS_list[order][1].split(",")[4].split(" ")[1][1:]) / float(int(GPS_list[order][1].split(",")[5].split(" ")[1][:-2])))
                    time = str(clock) + ":" + str(minute) + ":" + str(second)
                    GPS_list[order][1] = time
                if name2 == "GPSAltitude" :
                    alti = int(GPS_list[order][1].split(",")[0][1:]) / float(int(GPS_list[order][1].split(",")[1].split(" ")[1][:-1]))
                    GPS_list[order][1] = alti
                compare_value.append(GPS_list[order][1])
            compare_value = list(set(compare_value))
        else :
            print ("\nPlease Cheak the tag_name!")

    for order in range(0, len(GPS_list)) :
        if GPS_list[order][1] in compare_value :
            if len(compare_value) == 1 :
                fin_val_1.append(GPS_list[order][2])
                dic_tag_value[compare_value[0]] = fin_val_1
            elif len(compare_value) == 2 :
                if compare_value[0] == GPS_list[order][1] :
                    fin_val_1.append(GPS_list[order][2])

```

- IFD가 GPS인 조건일 때이며, 역할은 위와 동일하다.
- Excel File 생성 시, 지정한 GPSLatitude, GPSLongitude, GPSAltitude, GPSTimeStamp 형식을 분석에 용이하도록 가독성 좋게 변화 시켜 출력한다.

변환 수식은 앞서 설명한 것과 동일하다.

[illegible]

- IFD가 1st인 조건일 때이며, 역할은 위와 동일하다.

★ 참고 사항 : Interop의 경우 분석 대상에서 고려하지 않아도 부분이라 배제함.

임의적인 배제가 아닌 Excel 값을 본 후 판단.

```

else :
    print ("\nPlease, Input the right idf value\n")
    print ("idf key value : 0th, Exif, GPS, 1st")

if len(compare_value) == 1 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 2 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 3 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 4 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 5 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 6 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 7 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 8 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 9 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 10 :
    final_tag_value[name2] = dic_tag_value
elif len(compare_value) == 11 :
    final_tag_value[name2] = dic_tag_value
else :
    final_tag_value[name2] = dic_tag_value
print ("\nidf : " + name1 + ", " + name2 + "'s value\n")
print (final_tag_value[name2])
print ("\n")

```

- {Tag value : Filename}으로 생성한 dictionary를 다시 final_tag_value로 명명한 dictionary에 Tag name의 value 값으로 지정하고, 키는 사용자가 입력한 Tag name으로 하여 생성한다.
- 이 후, 사용자가 입력한 Tag name에 해당하는 value 값(Tag value : Filename)만 출력한다.

3. 분석 결과 (2.3의 내용과 비슷)

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
2	Orientation	1		1		6	6	1	6	1	1		1
3	GPSTag	10162	5940	5940	5940	5940	5940	10162	5940	10162	5940	5940	5940
4	DateTime	b'2018:01:09 10:54:25'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2018:01:09 11:16:52'	b'2017:05:23 13:26:11'	b'2018:01:09 11:16:25'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'
5	XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
6	Software	b'Microsoft Windows Photo Viewer 6.1.7600.16385'						b'Microsoft Windows Photo Viewer 6.1.7600.16385'		b'Microsoft Windows Photo Viewer 6.1.7600.16385'			
7	YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
8	Make	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'
9	Model	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'
10	ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
11	YCbCrPositioning	1	1	1	1	1	1	1	1	1	1	1	1
12	ExifTag	2326	184	195	184	195	195	2326	195	2326	195	184	195
13													

- 파일에 존재하는 IFD의 값으로 Excel에 Sheet가 생성됨을 확인 할 수 있다. 또한, A행 2열부터 각 행에는 Tag name이 입력 되어 있으며, B열부터 각 행 1열에는 파일명이 존재한다.

3.1 0th

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
Orientation	1		1		6	6	1	6	1	1		1
GPSTag	10162	5940	5940	5940	5940	5940	10162	5940	10162	5940	5940	5940
DateTime	b'2018:01:09 10:54:25'	b'2017:05:23 13:14:42'	b'2017:05:23 13:14:50'	b'2017:05:23 13:14:50'	b'2017:05:23 13:26:51'	b'2017:05:23 13:26:05'	b'2018:01:09 11:16:52'	b'2017:05:23 13:26:11'	b'2018:01:09 11:16:25'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:33'	b'2017:05:23 13:14:42'
XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Software	b'Microsoft Windows Photo Viewer 6.1.7600.16385'						b'Microsoft Windows Photo Viewer 6.1.7600.16385'		b'Microsoft Windows Photo Viewer 6.1.7600.16385'			
YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Make	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'	b'LG Electronics'
Model	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'	b'LGM-G600K'
ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
YCbCrPositioning	1	1	1	1	1	1	1	1	1	1	1	1
ExifTag	2326	184	195	184	195	195	2326	195	2326	195	184	195

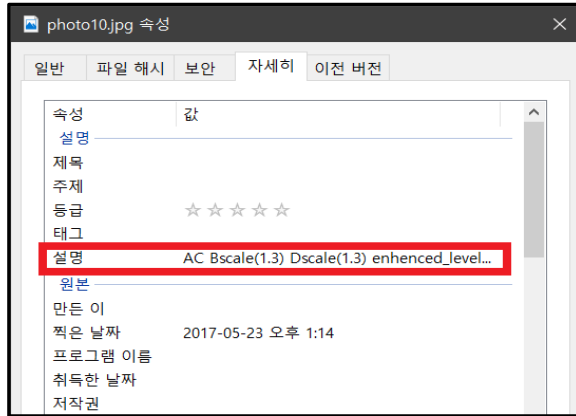
- 각 Tag name에 존재하는 value들 중 대체적인 값은 빨강, 상대적으로 적은 수의 일치 값들을 파랑으로 설정하면 위와 같은 Insight를 얻을 수 있다.
- GPSTag의 부분과 ExifTag 부분의 차이 발생은 해당 Tag들 사이에 존재하는 값들이 다른 그림 파일과 길이가 다르며, 사용자 임의 조작으로 인해 길이가 달라 질 수도 있다고 판단.
- 특정 파일만 software가 명시 되며, 다른 파일의 경우 의도적으로 삭제를 했을 수도 있다고 판단
- DateTime이 다른 3가지는 사용자가 시간을 조작 했을 수도 있다고 판단.

3.2 Exif

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
SubSecTimeDigitized	b'908354'	b'671598'	b'348982'	b'348982'	b'908354'	b'105920'	b'105920'	b'959989'	b'959989'	b'613968'	b'613968'	b'671598'
FileNumber	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)	(180, 100)
SceneCaptureType	0	0	0	0	0	0	0	0	0	0	0	0
SubSecTime	b'908354'	b'671598'	b'348982'	b'348982'	b'908354'	b'105920'	b'105920'	b'959989'	b'959989'	b'613968'	b'613968'	b'671598'
ExposureProgram	0	0	0	0	0	0	0	0	0	0	0	0
PixelDimension	2080	27633920	4160	27633920	4160	4160	2080	4160	2080	4160	27633920	4160
ColorSpace	1	1	1	1	1	1	1	1	1	1	1	1
FlashpixVersion	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'	b'0100'
ComponentsConfiguration	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x	b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x;b'0011w'02w'03w'x
DigitalZoomRatio	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)	(100, 100)
ISO Speed/Ratings	2700	50	50	50	2700	2700	2700	2700	2700	50	50	50

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
MeteringMode	2	2	2	2	2	2	2	2	2	2	2	2
	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' ASiCmm0w00w0 x00 AC Bscale(1.3) Dscale(1.3) enhanced_level(7)	b' AC Bscale(1.3) Dscale(1.3) enhanced_level(7)	b' ASiCmm0w00w0 x00 AC Bscale(1.3) Dscale(1.3) enhanced_level(7)	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' A1SHUTTER G547 IN3 N3 0040 Y6000	b' HB = 0.0 Fv=0 Fb = 0 outdoor = 0 ISO = 0	b' ASiCmm0w00w0 x00 HB = 0.0 Fv=0 Fb = 0 outdoor = 0 ISO = 0
	C600 Y71 CT1 x0 y8B00 x0 C100 FMO CR104	enhanced_level(7) isOutdoor(0) result(0) FMO	enhanced_level(7) isOutdoor(0) result(0) FMO	enhanced_level(7) isOutdoor(0) result(0) FMO	C600 Y71 CT1 x0 y8B00 x0 C100 FMO CR104	C600 Y71 CT1 x0 y8B00 x0 C100 FMO CR104	C600 Y71 CT1 x0 y8B00 x0 C100 FMO CR104	C600 Y71 CT1 x0 C100 FMO CR104	C600 Y71 CT1 x0 C100 FMO CR104	C600 Y71 CT1 x0 C100 FMO CR104	exp=0.000000 exp=0.000000	enhanced_level(7) isOutdoor(0) result(0) FMO CR104
	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC001000110:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC11111111:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	1635134821 AC Bscale(1.3) Dscale(1.3) enhanced_level(15)	return = FC001000110:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0
UserComment												
	024211bb11b1c 13 122611911194 13	03451151242 0 122611911194 13	0254519d149 4 b 122611911194 13	0254519d149 4 b 122611911194 13	024211bb11b1c 13 122611911194 13	024211bb11b1c 13 122611911194 13	024211bb11b1c 13 122611911194 13	024211bb11b1c 13 122611911194 13	024211bb11b1c 13 122611911194 13	024211bb11b1c 13 122611911194 13	FC000111011:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC000111011:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0
	0204117117c 12 0204115d151a 2	01a47195d12 2 d 122611911194 13	127850d612f 4 b 122611911194 13	127850d612f 4 b 122611911194 13	0204117117c 12 0204115d151a 2	0204117117c 12 0204115d151a 2	0204117117c 12 0204115d151a 2	0204117117c 12 0204115d151a 2	0204117117c 12 0204115d151a 2	0204117117c 12 0204115d151a 2	FC000111011:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0	FC000111011:bfalc 0000xxxxxxG600K1 Orzzzzzz14212532 8B04385F3184502 e0140 924 cb2a3 0 0346115242 0 0 0 0 0 0 0 0 0 0
	022b10661102 2 022b100c100b 2	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	022b10661102 2 022b100c100b 2	022b10661102 2 022b100c100b 2	022b10661102 2 022b100c100b 2	022b10661102 2 022b100c100b 2	022b10661102 2 022b100c100b 2	022b10661102 2 022b100c100b 2	034711517d 0 2 1196713d4d12 2 e	034711517d 0 2 1196713d4d12 2 e
	0259100e100d 21 0270100a100a 21	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	0259100e100d 21 0270100a100a 21	0259100e100d 21 0270100a100a 21	0259100e100d 21 0270100a100a 21	0259100e100d 21 0270100a100a 21	0259100e100d 21 0270100a100a 21	0259100e100d 21 0270100a100a 21	01677123f5e 4 a 01677123f5e 4 a	01677123f5e 4 a 01677123f5e 4 a
	0287100a100a 21 029a10091009 21	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	0287100a100a 21 029a10091009 21	0287100a100a 21 029a10091009 21	0287100a100a 21 029a10091009 21	0287100a100a 21 029a10091009 21	0287100a100a 21 029a10091009 21	0287100a100a 21 029a10091009 21	01677108f12 4 a 01677108f12 4 a	01677108f12 4 a 01677108f12 4 a
	02b05091009 21 02cc10091009 21	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	02b05091009 21 02cc10091009 21	02b05091009 21 02cc10091009 21	02b05091009 21 02cc10091009 21	02b05091009 21 02cc10091009 21	02b05091009 21 02cc10091009 21	02b05091009 21 02cc10091009 21	015a7009b0c 4 a 01467009b0c 4 a	015a7009b0c 4 a 01467009b0c 4 a
	02e310091009 21 02e310091009 21	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	127850d6125 4 b 127850d6125 4 b	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	02e310091009 21 02e310091009 21	016370609b 4 a 01570709b0c 4 a	016370609b 4 a 01570709b0c 4 a
SubSecTimeOriginal	b/908354	b/617598	b/349892	b/349892	b/908354	b/105920	b/105920	b/959989	b/959989	b/613968	b/613968	b/617598
ShutterSpeedValue	(2234, 1000)	(5321, 1000)	(5321, 1000)	(5321, 1000)	(2234, 1000)	(2234, 1000)	(2234, 1000)	(2234, 1000)	(2234, 1000)	(5321, 1000)	(5321, 1000)	(5321, 1000)
ExposureTime	(21, 100)	(1, 40)	(1, 40)	(1, 40)	(21, 100)	(21, 100)	(21, 100)	(21, 100)	(21, 100)	(1, 40)	(1, 40)	(1, 40)

- X, Y의 크기를 나타내는 부분에서 의심스러운 단위의 수가 입력되어 있으며, 기본적으로 (4160, 2080)이라 판단이 된다. X, Y 쌍이 변경 된 부분은 사진 촬영 시, 가로촬영 또는 세로촬영의 옵션 부분이라 생각 된다.
- DateTimeOriginal의 경우 사진이 찍힌 본래의 시간이라 판단되며, 앞선 IFD가 0th인 시간 부분과 비교를 하여 위,변조 여부를 판단이 가능하다.
- UserComment 부분에 3개의 파일만 ASCII라고 명시가 되어 있다. 해당 부분 파일의 속성을 확인 하였을 때, 설명 부분에 입력 값이 명시 된다.



3.3 GPS

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
GPSLatitudeRef	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'	b'N'
GPSLatitude	37.027794	37.026524	37.026449	37.026449	37.027794	37.027768	37.027768	37.02777	37.02777	37.026591	37.026591	37.026524
GPSAltitudeRef	0	0	0	0	0	0	0	0	0	0	0	0
GPSTimeStamp	4:1:49		4:1:49		4:1:49	4:1:2	4:1:2	4:1:10	4:1:10	4:1:31		4:1:41
GPSLongitude	127.024582	127.024653	127.024601	127.024601	127.024582	127.024596	127.024596	127.024594	127.024594	127.024498	127.024498	127.024653
GPSTimeStamp	b'2017:05:23'		b'2017:05:23'		b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'	b'2017:05:23'		b'2017:05:23'
GPSLongitudeRef	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'	b'E'
GPSAltitude	207.0	0.0	0.0	0.0	207.0	206.0	206.0	206.0	206.0	0.0	0.0	0.0

- 해당 파랑 부분을 보면 3개의 파일은 GPS가 기록 될 때의 날짜와 시간이 명시 되어 있지 않다. 애초 GPS 값이 없는 파일에 의도적으로 GPS 위,경도를 삽입 또는 사진 촬영 시, 기록된 GPS의 시간 값을 삭제 했을 수도 있다고 판단.
- 녹색의 경우 추가적으로 고려 할 수 있는 부분이라 판단하여 색상을 다르게 표시함. 실제로 촬영 시 기기에서 고도를 인식 못하여 기록이 안되었을 수도 있지만, 다른 파일과 비교 했을 때, 실제 값이 있는 파일도 존재하는 것으로 보아 0.0으로 기록된 값은 위,변조가 되었을 수도 있다는 가능성이 존재. (상대적으로 중요하게 고려할 단계의 Tag라 판단은 하지 않음)

3.4 Interop

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
InteroperabilityIndex	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'	b'R98'

- 의견 없음.

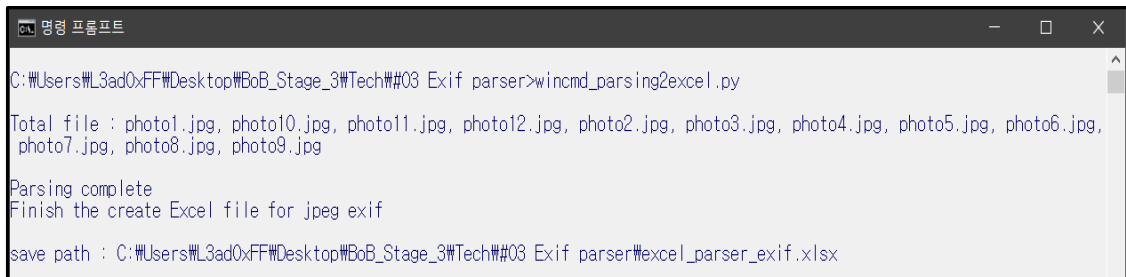
3.5 1st

Tag / Filename	photo1.jpg	photo10.jpg	photo11.jpg	photo12.jpg	photo2.jpg	photo3.jpg	photo4.jpg	photo5.jpg	photo6.jpg	photo7.jpg	photo8.jpg	photo9.jpg
Orientation	6	1	1	1	6	6	6	6	6	1	1	1
XResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
Compression	6	6	6	6	6	6	6	6	6	6	6	6
ResolutionUnit	2	2	2	2	2	2	2	2	2	2	2	2
YResolution	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)	(72, 1)
JPEGInterchangeFormatLength	3341	4044	16441	5109	2819	2801	3334	3907	4412	15167	4533	13532
JPEGInterchangeFormat	10464	6180	6239	6180	6239	6239	10464	6239	10464	6239	6180	6239

- Thumbnail과 관련 있는 tag로 판단하며, 의견 없음.

3.6 실행 내역

3.6.1 실행 경로에 있는 파일 search 후, EXIF2Excel 저장



```

C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>wincmd_parsing2excel.py

Total file : photo1.jpg, photo10.jpg, photo11.jpg, photo12.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg, photo6.jpg,
photo7.jpg, photo8.jpg, photo9.jpg

Parsing complete
Finish the create Excel file for jpeg exif

save path : C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser\excel_parser_exif.xlsx

```

3.6.2 0th

- 선정한 Tag Name : DateTime, Software, GPSTag, ExifTag

3.6.2.1 DateTime



```

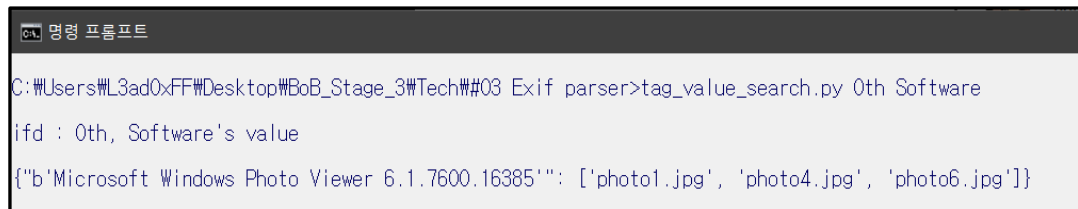
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py 0th DateTime

ifd : 0th, DateTime's value

{'b'2018:01:09 10:54:25'': ['photo1.jpg', 'photo10.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo4.jpg'], "b'2017:05:23 13:14:42'": ['photo12.jpg', 'photo9.jpg'], "b'2017:05:23 13:14:50'": ['photo12.jpg', 'photo9.jpg'], "b'2017:05:23 13:26:51'": ['photo1.jpg', 'photo10.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo4.jpg'], "b'2017:05:23 13:26:05'": ['photo1.jpg', 'photo10.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo4.jpg'], "b'2018:01:09 11:16:52'": ['photo1.jpg', 'photo10.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo4.jpg'], "b'2017:05:23 13:26:11'": ['photo5.jpg', 'photo7.jpg', 'photo8.jpg'], "b'2018:01:09 11:16:25'": ['photo6.jpg'], "b'2017:05:23 13:14:33'": ['photo5.jpg', 'photo7.jpg', 'photo8.jpg']}

```

3.6.2.2 Software



```

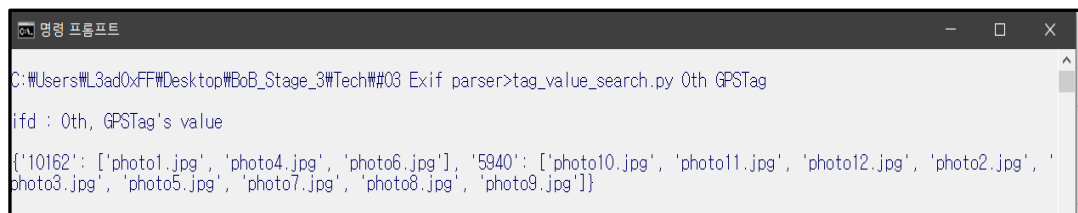
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py 0th Software

ifd : 0th, Software's value

{'b'Microsoft Windows Photo Viewer 6.1.7600.16385'': ['photo1.jpg', 'photo4.jpg', 'photo6.jpg']}

```

3.6.2.3 GPSTag



```

C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py 0th GPSTag

ifd : 0th, GPSTag's value

{'10162': ['photo1.jpg', 'photo4.jpg', 'photo6.jpg'], '5940': ['photo10.jpg', 'photo11.jpg', 'photo12.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo8.jpg', 'photo9.jpg']}

```

3.6.2.4 ExifTag



```

C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py 0th ExifTag

ifd : 0th, ExifTag's value

{'2326': ['photo1.jpg', 'photo4.jpg', 'photo6.jpg'], '184': ['photo10.jpg', 'photo12.jpg', 'photo8.jpg'], '195': ['photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo9.jpg']}

```

3.6.3 Exif

- 선정한 Tag Name : PixelXDimension, PixelYDimension, DateTimeOriginal, UserComment


3.6.3.1 PixelXDimension

```
C:\Users\3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py Exif PixelXDimension

ifd : Exif, PixelXDimension's value

{'2080': ['photo1.jpg', 'photo4.jpg', 'photo6.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo9.jpg'], '272633920': ['photo10.jpg', 'photo12.jpg', 'photo8.jpg'], '4160': ['photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo9.jpg', 'photo1.jpg', 'photo4.jpg', 'photo6.jpg']}
```

3.6.3.2 PixelYDimension



```
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\03 Exif parser>tag_value_search.py Exif PixelYDimension

lfd : Exif, PixelYDimension's value

{'2080': ['photo1.jpg', 'photo4.jpg', 'photo6.jpg', 'photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo9.jpg'], '4160': ['photo11.jpg', 'photo2.jpg', 'photo3.jpg', 'photo5.jpg', 'photo7.jpg', 'photo9.jpg', 'photo1.jpg', 'photo4.jpg', 'photo6.jpg'], '136316960': ['photo10.jpg', 'photo12.jpg', 'photo8.jpg']}
```

3.6.3.3 DateTimeOriginal



```
C:\Users\ML3ad0xFF\Desktop\BoB_Stage_3\Tech\#03 Exif parser>tag_value_search.py Exif DateTimeOriginal

lfd : Exif, DateTimeOriginal's value

{'b'2017:05:23 13:26:51': ['photo1.jpg', 'photo2.jpg', 'photo1.jpg', 'photo2.jpg'], "b'2017:05:23 13:14:42'": ['photo10.jp
g', 'photo9.jpg', 'photo10.jpg', 'photo9.jpg'], "b'2017:05:23 13:14:50'": ['photo11.jpg', 'photo12.jpg', 'photo11.jpg', 'ph
oto12.jpg'], "b'2017:05:23 13:26:05'": ['photo3.jpg', 'photo4.jpg', 'photo3.jpg', 'photo4.jpg'], "b'2017:05:23 13:26:11'":
['photo5.jpg', 'photo6.jpg', 'photo5.jpg', 'photo6.jpg'], "b'2017:05:23 13:14:33'": ['photo7.jpg', 'photo8.jpg', 'photo7.jp
g', 'photo8.jpg']}
```

3.6.3.4 UserComment

[illegible]

- UserComment의 경우 string길이가 길어서 한번에 나오지는 않음.

3.6.4 GPS

- 선정한 Tag Name : GPSLatitude, GPSLongitude, GPSTimeStamp, GPSAltitude

3.6.4.1 GPSLatitude

```
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\##03 Exif parser>tag_value_search.py GPS GPSLatitude

ifd : GPS, GPSLatitude's value

{37.027794: ['photo1.jpg', 'photo2.jpg'], 37.026524: ['photo10.jpg', 'photo9.jpg'], 37.026449: ['photo11.jpg', 'photo12.jpg'], 37.027768: ['photo3.jpg', 'photo4.jpg'], 37.02777: ['photo5.jpg', 'photo6.jpg'], 37.026591: ['photo7.jpg', 'photo8.jpg']}
```

3.6.4.2 GPSLongitude

```
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\##03 Exif parser>tag_value_search.py GPS GPSLongitude

ifd : GPS, GPSLongitude's value

{127.024582: ['photo1.jpg', 'photo2.jpg'], 127.024653: ['photo10.jpg', 'photo9.jpg'], 127.024601: ['photo11.jpg', 'photo12.jpg'], 127.024596: ['photo3.jpg', 'photo4.jpg'], 127.024594: ['photo5.jpg', 'photo6.jpg'], 127.024498: ['photo7.jpg', 'photo8.jpg']}
```

3.6.4.3 GPSTimeStamp

```
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\##03 Exif parser>tag_value_search.py GPS GPSTimeStamp

ifd : GPS, GPSTimeStamp's value

{'4:1:49': ['photo1.jpg', 'photo11.jpg', 'photo2.jpg'], '4:1:2': ['photo3.jpg', 'photo4.jpg'], '4:1:10': ['photo5.jpg', 'photo6.jpg'], '4:1:31': ['photo7.jpg'], '4:1:41': ['photo9.jpg']}
```

3.6.4.4 GPSAltitude

```
C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\##03 Exif parser>tag_value_search.py GPS GPSTimeStamp

ifd : GPS, GPSTimeStamp's value

{'4:1:49': ['photo1.jpg', 'photo11.jpg', 'photo2.jpg'], '4:1:2': ['photo3.jpg', 'photo4.jpg'], '4:1:10': ['photo5.jpg', 'photo6.jpg'], '4:1:31': ['photo7.jpg'], '4:1:41': ['photo9.jpg']}

C:\Users\L3ad0xFF\Desktop\BoB_Stage_3\Tech\##03 Exif parser>tag_value_search.py GPS GPSAltitude

ifd : GPS, GPSAltitude's value

{207.0: ['photo1.jpg', 'photo2.jpg'], 0.0: ['photo10.jpg', 'photo11.jpg', 'photo12.jpg', 'photo7.jpg', 'photo8.jpg', 'photo9.jpg'], 206.0: ['photo3.jpg', 'photo4.jpg', 'photo5.jpg', 'photo6.jpg']}
```

4. 결론

4.1 Exif.DateTimeOriginal과 0th.DateTime과의 비교

- 'Exif.DateTimeOriginal : 2017:05:23'과 0th.DateTime이 일치하는 대부분의 파일이 존재하지만, photo1.jpg, photo4.jpg, photo6.jpg 파일은 두 TagValue가 다르다.
- [photo1.jpg, photo4.jpg, photo6.jpg]를 하나의 그룹(1번 그룹)과 나머지 파일을 하나의 그룹으로 하여 비교 시, 1번 그룹에만 software 정보가 존재한다.
- Group 1 : [photo1.jpg, photo4.jpg, photo6.jpg]
Group 2 : [Photo10.jpg, photo11.jpg, photo12.jpg, photo2.jpg, photo3.jpg, photo5.jpg, photo7.jpg, photo8.jpg, photo9.jpg]
- 위와 같이 그룹 분류 후 분석 시, 0th.GPSTag, 0th.Software, 0th.ExifTag, Exif.InteroperabilityTag와 같은 Tag name의 값이 상이함을 확인 가능

4.1.1 0th.GPSTag

Group 1 : 10162 / Group 2 : 5940

4.1.2 0th.Software

Group 1 : Microsoft Windows Photo Viewer 6.1.7600.16385 / Group 2 : None

4.1.3 0th.ExifTag

Group 1 : 2326 / Group 2 : 195

4.1.4 Exif.InteroperabilityTag

Group 1 : 10120 / Group 2 : 2080

4.2 Exif.PixelXDimension & Exif.PixelYDimension

- 그림의 크기를 결정하는 요소이며, photo10.jpg, photo12.jpg, photo8.jpg의 값이 다른 파일들에 비해 큰 차이를 보이고 있다.
- [photo10.jpg, photo12.jpg, photo8.jpg](그룹 1번)과 다른 파일로 그룹을 나눌 수 있으며, 이때 정의한 1번 그룹에 속한 파일들은 파일 속성에서 보이는 값 (4160, 2080)과 다름을 확인할 수 있다

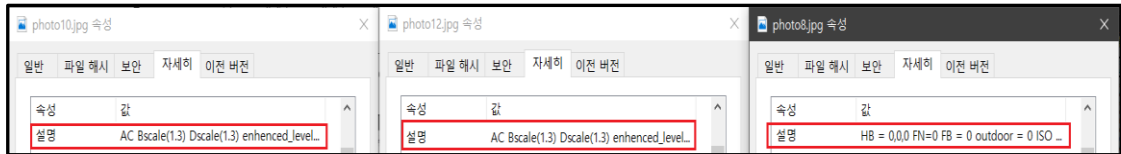


Group 1 : [Photo10.jpg, photo12.jpg, photo8.jpg,]

Group 2 : [photo1.jpg,, photo11.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg,
photo6.jpg, photo7.jpg, , photo9.jpg]

4.3 Exif.UserComment

- 해당 부분에 ASCII가 명시되어 다음에 올 string을 파일의 속성에 표시한다.



- 해당 내용으로도 [photo10.jpg, photo12.jpg]와 [photo8.jpg]로 분류가 가능하다
- Group 1 : [[Photo10.jpg, photo12.jpg], photo8.jpg,] – ASCII로 명시
Group 2 : [photo1.jpg,, photo11.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg,
photo6.jpg, photo7.jpg, , photo9.jpg]. – ASCII 문자열 없음.

4.4 GPSTimeStamp & GPSDateStamp

- 모든 파일에 GPSLatitude와 GPSLongitude가 존재한다. 그러나 추가 GPSTimeStamp와 GPSDateStamp를 확인 하면 2개의 Group으로 분류가 가능하다.
- 3.6.4.3과 3.6.4.4의 항목에 제시한 분류 참조
- Group 1 : [Photo10.jpg, photo12.jpg, photo8.jpg,]
Group 2 : [photo1.jpg,, photo11.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg,
photo6.jpg, photo7.jpg, , photo9.jpg]

4.5 최종 결론

- 위의 의견을 종합하였을 때, [photo10.jpg, photo12.jpg, photo8.jpg]가 위, 변조가 되었을 것이라고 가능성이 높다고 판단한다.
- 총12개의 분석 그림파일을 아래와 같이 분류 한다.
- Group 1 : [Photo10.jpg, photo12.jpg, photo8.jpg,]
Group 2 : [photo1.jpg,, photo11.jpg, photo2.jpg, photo3.jpg, photo4.jpg, photo5.jpg,
photo6.jpg, photo7.jpg, , photo9.jpg]
Group 2-1 : [photo11.jpg, photo2.jpg, photo3.jpg, photo5.jpg, photo7.jpg, , photo9.jpg]
Group 2-2 : [photo1.jpg, photo4.jpg, photo6.jpg]
(Group 2의 세부 분류는 4.1의 결과를 따른 결론이다.)

5. 참고문헌

- Standard of Japan Electronics and Information Technology Industries Association, 'Exchangeable image file format for digital still cameras : Exif Version 2.2'
- Sungjin Hong & Kyung-Hyune Rhee(Pukyong National University), An approach for similar file detection with GPS information
- 박재유, [파이썬 포렌식] 이미지 GPS 정보 추출하기, <https://cpuu.postype.com/post/23100>