

# **Laporan Projek UAS Georeferenced Image Stitching**



## **Disusun Oleh :**

Nama : Sanvic Dicaprio  
NIM : 09011282227081  
Kelas : SK5C

## **Dosen Pengampu :**

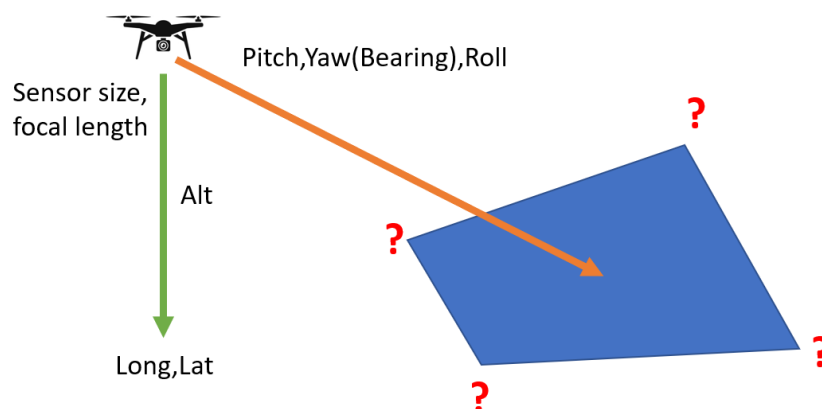
Adi Hermansyah, S.Kom., M.T.

**Fakultas Ilmu Komputer  
Jurusan Sistem Komputer  
Universitas Sriwijaya Tahun 2023-2024**

## Latar Belakang Proyek

*Image stitching* merupakan proses menggabungkan beberapa gambar menjadi satu gambar panorama. Sementara GIS (*Georeferenced Image Stitching*) adalah *image stitching* yang menggunakan informasi tambahan yaitu GPS pada data gambar tersebut. Program pada GIS memiliki kemampuan untuk dapat memperhitungkan nilai spasial seperti koordinat lintang (*latitude*), bujur (*longitude*), dan ketinggian (*altitude*) yang tersimpan dalam format EXIF (*Exchangeable Image File Format*). Biasanya proses ini melibatkan sumber gambar yang diambil dari ketinggian tertentu dengan menggunakan alat bantu seperti drone. Hasil dari *image stitching* ini yaitu sebuah panorama yang memiliki visual secara vertikal dan menyerupai lokasi aslinya.

GIS memiliki peran yang luas dalam aplikasi pemetaan gambar digital, survei lapangan, dan peranan berbasis lokasi lainnya. Pemanfaatan data GPS memungkinkan panorama yang diproses *image stitching* digunakan dalam konteks geografis yang lebih luas dan membuka potensi untuk pemahaman mendalam tentang lokasi dalam format visual. Beberapa bidang yang telah diketahui menggunakan GIS yaitu bidang pertanian, infrastruktur dan konstruksi, serta teknologi informasi satelit. Oleh karena itu, GIS membutuhkan teknik-teknik khusus untuk menangan tantangan seperti perbedaan sudut pandang, distorsi geometris, variasi pencahayaan, dan ketidaksesuaian struktur pada gambar.



Gambar 1. Visual penggunaan drone untuk mengambil gambar dengan detail GPS

## Tujuan Proyek

Tujuan pembuatan proyek kali ini yaitu :

1. Untuk mengetahui bentuk panorama yang dibuat dengan informasi GPS dalam gambar input.
2. Untuk mengetahui perbedaan antara panorama biasa dengan panorama dari GIS.

## Metode Proyek

Proyek ini dibuat dengan menggunakan pemrograman python yang mengekstrak informasi GPS pada EXIF dengan library piexif. Pertama, program akan membaca informasi dari nilai latitude, longitude, dan altitude foto. Jika foto memiliki ketiga informasi GPS, program akan langsung merangkai gambar-gambar tersebut memakai

algoritma image stitching dengan bantuan library cv2 dan glob. Hasil akhir diperkirakan akan membentuk panorama dengan menampilkan informasi GPS sesuai panorama. Program dijalankan secara paralel menggunakan MPI dengan 3 VM (*Virtual Machine*) dari Oracle VirtualBox berbasis GUI (*Graphical User Interface*) dengan spesifikasi VM sebagai berikut :

- Jenis sistem operasi : Zorin OS 16.3 x86\_64
- Processor : 12<sup>th</sup> Gen Intel i5-12500H 2 Core 3.110GHz
- RAM Virtual : 1024 MB
- Jaringan : Host-Only Adapter

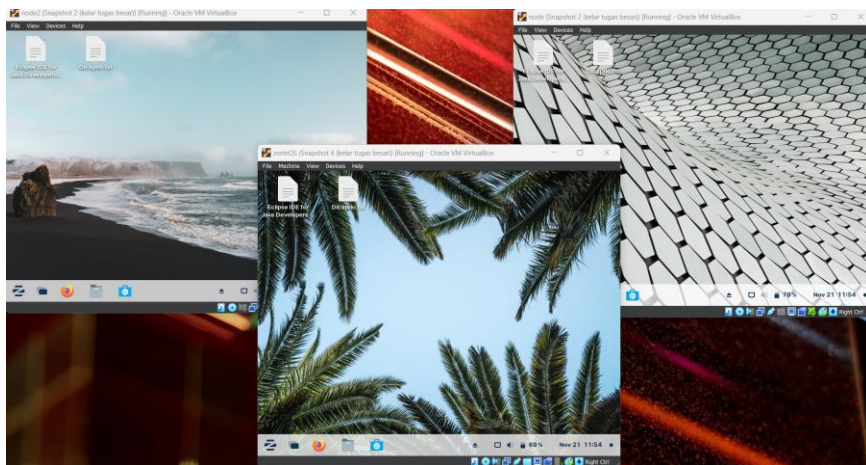
```
1 from mpi4py import MPI
2 import cv2
3 import argparse
4 import glob
5 import piexif
6 import numpy as np
7
8 cv2.occl.setUseOpenCL(False)
9
10 def get_lat_lon_alt(exif):
11     if 'GPS' not in exif:
12         return None, None, None
13
14     lat_tuple = exif['GPS'][piexif.GPSIFD.GPSLatitude]
15     lon_tuple = exif['GPS'][piexif.GPSIFD.GPSLongitude]
16     alt_tuple = exif['GPS'][piexif.GPSIFD.GPSAltitude]
17
18     lat = convert_to_degrees(lat_tuple)
19     lon = convert_to_degrees(lon_tuple)
20     alt = alt_tuple[0] / alt_tuple[1]
21
22     lat_ref = exif['GPS'][piexif.GPSIFD.GPSLatitudeRef]
23     lon_ref = exif['GPS'][piexif.GPSIFD.GPSLongitudeRef]
24
25     lat = lat * (-1 if lat_ref.upper() == 'S' else 1)
26     lon = lon * (-1 if lon_ref.upper() == 'W' else 1)
27
28     return lat, lon, alt
29
30 def convert_to_degrees(value):
31     d, m, s = value
32     return d[0] / d[1] + m[0] / (m[1] * 60) + s[0] / (s[1] * 3600)
33
34 def process_image(imagePath):
35     image = cv2.imread(imagePath)
36     exif_data = piexif.load(imagePath)
37     lat, lon, alt = get_lat_lon_alt(exif_data)
38
39     if lat is not None and lon is not None:
40         return image, lat, lon, alt, exif_data
41     else:
42         return None
43
44 def load_images_with_gps_parallel(imagePaths):
45     images_with_gps = []
46
47     comm = MPI.COMM_WORLD
48     rank = comm.Get_rank()
49     size = comm.Get_size()
50
51     chunk_size = len(imagePaths) // size
52     start_index = rank * chunk_size
53     end_index = start_index + chunk_size if rank != size - 1 else len(imagePaths)
54
55     local_image_paths = imagePaths[start_index:end_index]
56     local_results = []
57
58     for imagePath in local_image_paths:
59         result = process_image(imagePath)
60         if result is not None:
61             local_results.append(result)
62
63     all_results = comm.gather(local_results, root=0)
64
65     if rank == 0:
66         images_with_gps = [item for sublist in all_results for item in sublist]
67
68     return images_with_gps
69
70 def stitch_images(images):
71     num_images = len(images)
72
73     if num_images < 2:
74         print("[INFO] Insufficient images for stitching.")
```

```

74         print("[INFO] Insufficient images for stitching.")
75         return None
76
77     images_to_stitch = [img for img, _, _, _ in images]
78
79     if cv2.__version__.startswith('3'):
80         stitcher = cv2.createStitcher()
81     else:
82         stitcher = cv2.Stitcher_create()
83
84     (status, stitched) = stitcher.stitch(images_to_stitch)
85
86     if status == cv2.Stitcher_OK:
87
88         aspect_ratio = images[0][0].shape[1] / images[0][0].shape[0]
89         new_width = int(stitched.shape[0] * aspect_ratio)
90         stitched_resized = cv2.resize(stitched, (new_width, stitched.shape[0]))
91
92         return stitched_resized
93     else:
94         print(f"[INFO] Image stitching failed with status {status}")
95
96         return None
97
98 def main():
99     ap = argparse.ArgumentParser()
100     ap.add_argument("-i", "--images", type=str, required=True,
101                     help="path to input directory of images to stitch")
102     ap.add_argument("-o", "--output", type=str, required=True,
103                     help="path to the output image")
104     args = vars(ap.parse_args())
105
106     comm = MPI.COMM_WORLD
107     rank = comm.Get_rank()
108
109     if rank == 0:
110         print("[INFO] Memuat gambar...")
111
112     imagePaths = glob.glob(args["images"] + "/*.jpg")
113     images_with_gps = load_images_with_gps_parallel(imagePaths)
114
115     if rank == 0:
116         if not images_with_gps:
117             print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
118         else:
119             print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
120             images = [(img, lat, lon, alt, exif_data) for img, lat, lon, alt, exif_data in images_with_gps]
121             print("[INFO] Merangkai gambar...")
122
123             stitched_image = stitch_images(images)
124
125             if stitched_image is not None:
126                 cv2.imwrite(args["output"], stitched_image)
127
128                 output_exif_data = piexif.load(args["output"])
129                 for _, _, _, _, exif_data in images:
130                     if 'GPS' in exif_data:
131                         output_exif_data['GPS'] = exif_data['GPS']
132                     exif_bytes = piexif.dump(output_exif_data)
133                     piexif.insert(exif_bytes, args["output"])
134
135             print("[INFO] Penggabungan gambar berhasil.")
136         else:
137             print("[INFO] Penggabungan gambar gagal.")
138
139 if __name__ == "__main__":
140     main()
141

```

Gambar 2. Program python yang akan dijalankan dalam proyek GIS



Gambar 3. VM yang akan dipakai dalam proyek GIS

Untuk objek panorama, kami mengambil 18 foto dari lokasi Fasilkom Universitas Sriwijaya yang menampilkan informasi GPS. Objek diambil menggunakan drone secara vertikal dengan ketinggian tertentu.

| Image                 |                               |
|-----------------------|-------------------------------|
| Image ID              | 0457221C34617BEA295BFE0B05... |
| Dimensions            | 4608 x 3456                   |
| Width                 | 4608 pixels                   |
| Height                | 3456 pixels                   |
| Horizontal resolution | 72 dpi                        |
| Vertical resolution   | 72 dpi                        |
| Bit depth             | 24                            |
| Compression           |                               |
| Resolution unit       | 2                             |
| Color representation  | Uncalibrated                  |

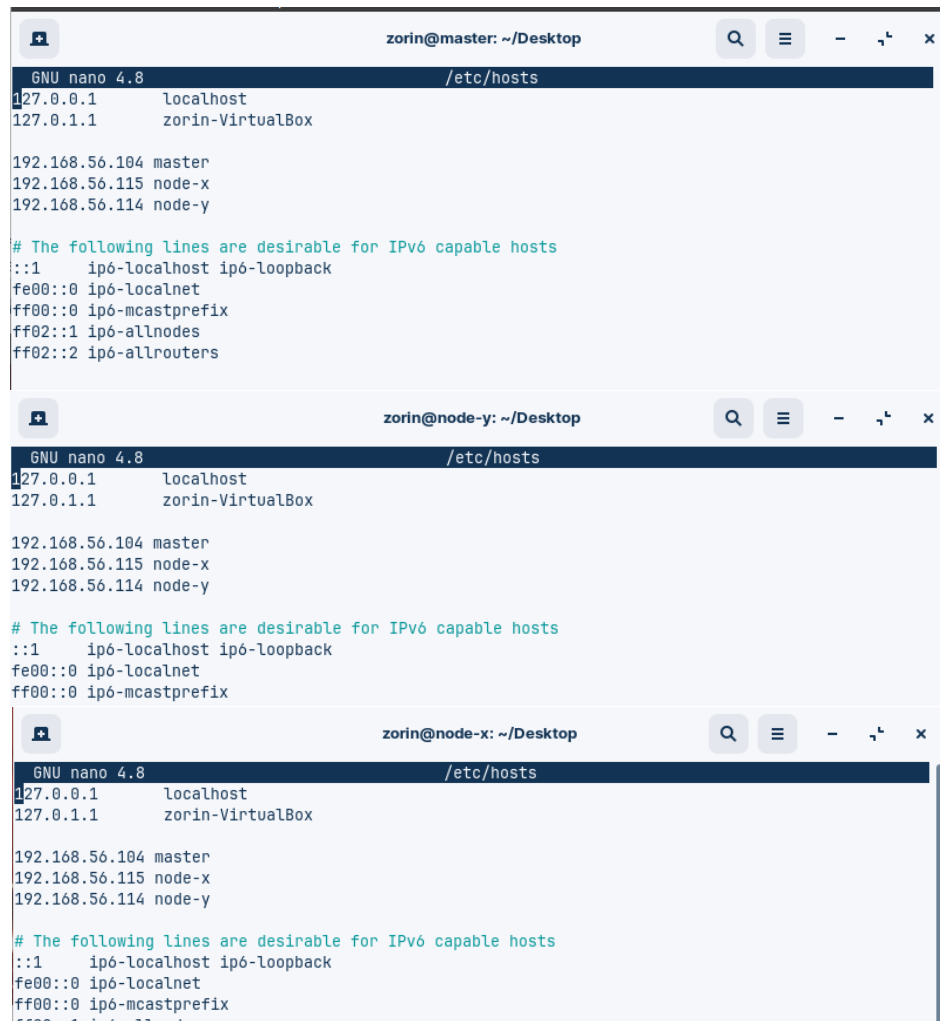
Gambar 4. Contoh detail fisik pada foto

| GPS       |                             |
|-----------|-----------------------------|
| Latitude  | 46; 31; 15.3147688704192575 |
| Longitude | 6; 32; 59.0515705761282561  |
| Altitude  | 505.681131651170404         |

Gambar 5. Contoh detail GPS pada foto

## Praktikum Proyek

1. Hidupkan ketiga VM dan masuk sebagai user MPI. User yang dipakai wajib sama antar 1 dengan VM lainnya. Hubungkan ketiga VM dengan menambah isi dari file “/etc/hosts” dengan nama hostname dan IP jaringan. Pastikan ketiga VM tersambung pada jaringan yang sama.



Gambar 6. Tampilan dari ketiga VM dalam file “/etc/hosts”

2. Install `openssh-server` di semua VM dengan “`sudo apt install openssh-server`”. Beralih ke master (master disini yaitu `zorin@master`), selesaikan set up keygen dan masuk sebagai localhost. Sambungkan ketiga VM dengan “`ssh-copy-id <nama_user>@<nama_hostname>`”.



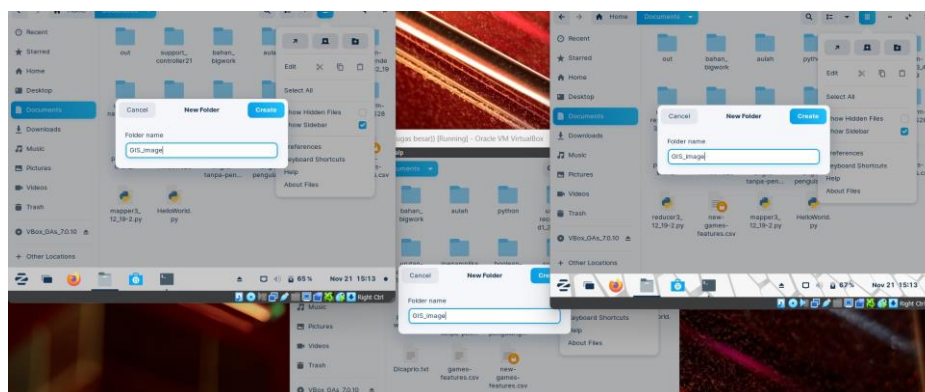
```
zorin@master: ~  
zorin@master:~$ ssh-copy-id zorin@master  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/zorin/.ssh/id_rsa.pub"  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
  
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.  
                (if you think this is a mistake, you may want to use -f option)  
  
zorin@master:~$ ssh-copy-id zorin@node-x  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/zorin/.ssh/id_rsa.pub"  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
  
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.  
                (if you think this is a mistake, you may want to use -f option)  
  
zorin@master:~$ ssh-copy-id zorin@node-y  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/zorin/.ssh/id_rsa.pub"  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
  
/usr/bin/ssh-copy-id: WARNING: All keys were skipped because they already exist on the remote system.  
                (if you think this is a mistake, you may want to use -f option)  
  
zorin@master:~$
```

Gambar 7. Tampilan saat berhasil masuk sebagai localhost dan copy id ssh hostname di terminal

3. Cek apakah mpi telah berjalan secara paralel dengan mengetik perintah “`mpirun -n <jumlah_komputer> -host <nama_host1>,<nama_host2>,<nama_host3> python3 -m mpi4py.bench helloworld`” (jumlah host tidak terlalu bergantung pada jumlah komputer. Kita dapat mengurangi jumlah komputer dibawah jumlah host namun tak dapat mengurangi jumlah host dibawah jumlah komputer).

```
zorin@master:~$ mpirun -n 3 -host master,node-x,node-y python3 -m mpi4py.bench helloworld  
Hello, World! I am process 0 of 3 on master.  
Hello, World! I am process 1 of 3 on node-x.  
Hello, World! I am process 2 of 3 on node-y.  
zorin@master:~$
```

4. Install layanan nfs agar bisa membagikan folder dengan “`sudo apt install nfs-kernel-server`”. Buatlah folder pada masing-masing VM dengan nama dan direktori yang sama.



5. Beralih ke terminal master, ketik “`sudo nano /etc/exports`”. Tambahkan kalimat dibawahnya dengan “<lokasi folder> \*(rw, sync, no\_root\_squash, no\_subtree\_check)”.



```
zorin@master: ~  
GNU nano 4.8 /etc/exports Modified  
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients. See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)  
#  
# Example for NFSv4:  
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)  
#  
/home/python *(rw,sync,no_root_squash,no_subtree_check)  
/home/zorin/Documents/aulah *(rw,sync,no_root_squash,no_subtree_check)  
/home/zorin/Documents/bahan_bigwork *(rw,sync,no_root_squash,no_subtree_check)  
/home/zorin/Documents/GIS_image *(rw,sync,no_root_squash,no_subtree_check)
```

6. Simpan dan kembali. Ketik “sudo exportfs -a” untuk mengekspor ulang semua folder yang ingin dibagikan. Setelah itu, ketik “sudo systemctl restart nfs-kernel-server” untuk mereset sistem service nfs.

```
zorin@master: ~$ sudo nano /etc/exports  
[sudo] password for zorin:  
zorin@master:~$ sudo exportfs -a  
zorin@master:~$ sudo systemctl restart nfs-kernel-server  
zorin@master:~$
```

7. Kembali pada 2 VM lain. Ketik “sudo mount <Hostname\_Master>:<lokasi folder di master> <lokasi folder di VM>”.

```
zorin@node-y: ~/Desktop  
zorin@node-y:~/Desktop$ sudo mount master:/home/zorin/Documents/GIS_image /home/zorin/Documents/GIS_image/  
zorin@node-y:~/Desktop$  
  
zorin@node-x: ~/Desktop  
zorin@node-x:~/Desktop$ sudo mount master:/home/zorin/Documents/GIS_image /home/zorin/Documents/GIS_image/  
zorin@node-x:~/Desktop$
```

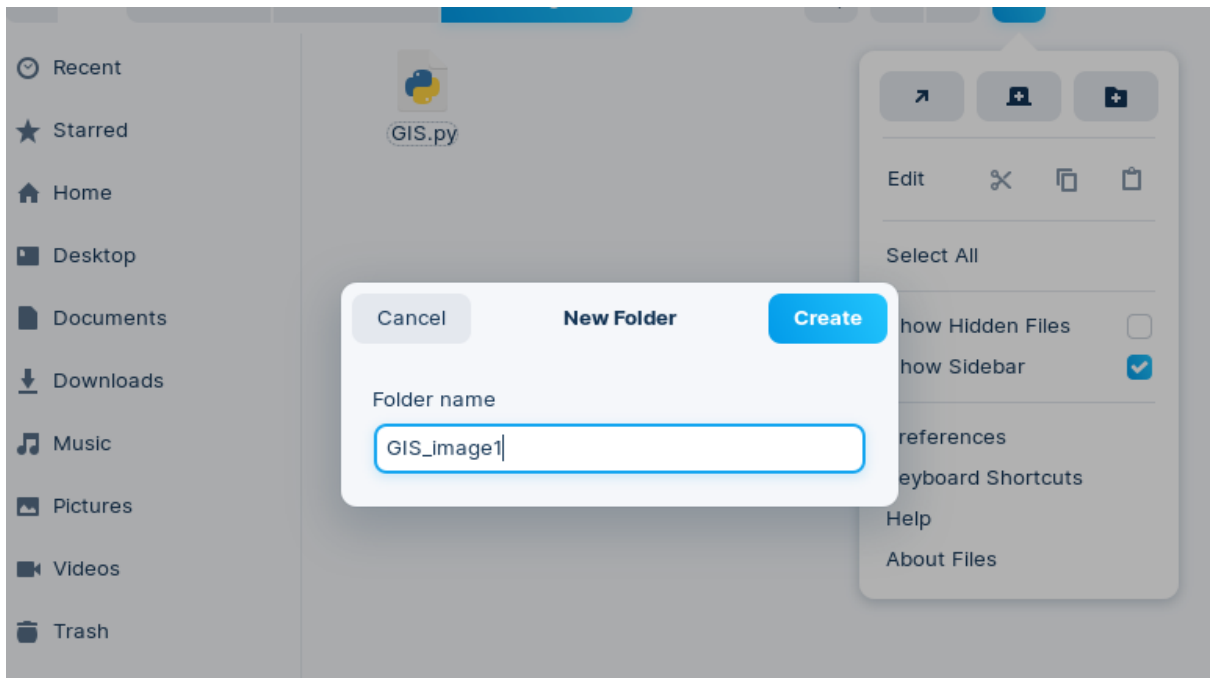
Gambar 8. Menghubungkan folder yang dibagikan master melalui layanan NFS

8. Cek dengan membuat file pada salah satu VM. Jika berhasil, maka akan tampil pada VM lain. (disini saya mengimpor codingan proyek GIS dengan nama GIS.py)

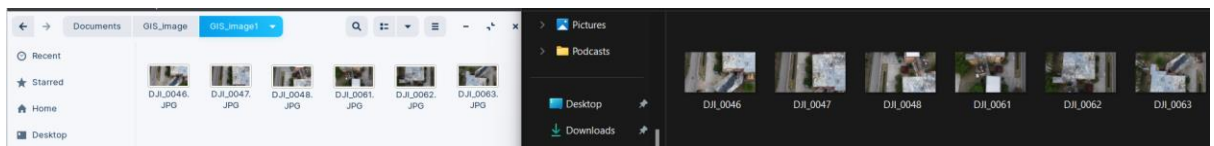


9. Buat folder yang akan menampung gambar proyek GIS 1.

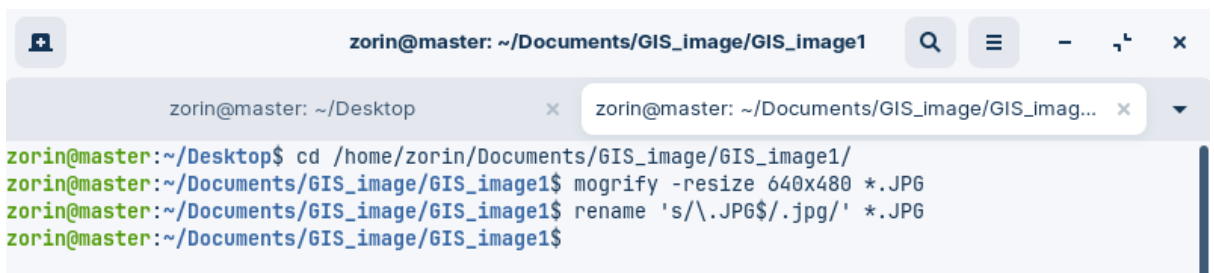




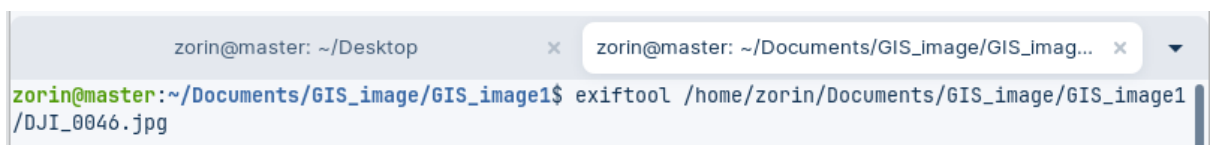
10. Ambil semua gambar dengan cara “drag and drop” semua gambar ke salah satu VM.



11. Samakan semua resolusi foto untuk memperkecil kemungkinan format yang error. Setelah itu, ubah format foto dari \*.JPG menjadi \*.jpg.



12. Untuk memastikan apakah data GPS dalam EXIF ada, kita dapat memakai tool exif dengan mengetik “exiftool <nama\_gambar>”



Gambar 13. Tool untuk mengetahui data EXIF

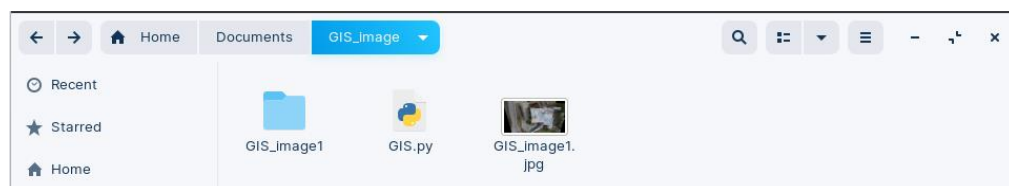
```
zorin@master: ~/Desktop x zorin@master: ~/Documents/GIS_image/GIS_imag... x
Has Settings : False
Has Crop : False
Already Applied : False
Image Width : 640
Image Height : 360
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:2 (2 1)
Aperture : 2.8
Image Size : 640x360
Megapixels : 0.230
Scale Factor To 35 mm Equivalent: 5.3
Shutter Speed : 1/320
Thumbnail Image : (Binary data 10255 bytes, use -b option to extract)
GPS Altitude : 133.7 m Above Sea Level
GPS Latitude : 3 deg 13' 13.10" S
GPS Longitude : 104 deg 39' 4.07" E
Circle Of Confusion : 0.006 mm
Field Of View : 73.7 deg
Focal Length : 4.5 mm (35 mm equivalent: 24.0 mm)
GPS Position : 3 deg 13' 13.10" S, 104 deg 39' 4.07" E
Hyperfocal Distance : 1.28 m
Light Value : 11.3
zorin@master:~/Documents/GIS_image/GIS_image1$
```

Gambar 14. Perintah exiftool setelah dieksekusi

13. Jika semua telah dilakukan, jalankan program python. Pastikan semua VM tetap aktif saat memproses GIS melalui MPI.

```
zorin@master:~/Desktop$ mpirun -n 3 -host master,node-x,node-y python3 /home/zorin/Documents/GIS_image/GIS.py --images /home/zorin/Documents/GIS_image/GIS_image1 --output /home/zorin/Documents/GIS_image/GIS_image1.jpg
[INFO] Memuat gambar...
[INFO] Merangkai gambar...
[INFO] Penggabungan gambar berhasil.
zorin@master:~/Desktop$
```

14. Ketika selesai, cek pada direktori yang telah ditentukan output. Lakukan pada input “GIS\_image2” dan “GIS\_image3”.



Gambar 15. Perintah MPI setelah dieksekusi



Gambar 16. Hasil gambar dari “GIS\_image1.jpg”



Gambar 17. Hasil gambar dari “GIS\_image2.jpg”



Gambar 18. Hasil gambar dari "GIS\_image3.jpg"

15. Periksa kembali apakah output pada gambar mengembalikan detail GPS dengan perintah yang sama seperti langkah 12.

```
File Size           : 314 kB
File Modification Date/Time : 2023:11:27 14:44:53+07:00
File Access Date/Time   : 2023:11:27 14:45:11+07:00
File Inode Change Date/Time : 2023:11:27 14:44:53+07:00
File Permissions       : rw-rw-r--
File Type             : JPEG
File Type Extension    : jpg
MIME Type             : image/jpeg
Exif Byte Order        : Big-endian (Motorola, MM)
GPS Version ID         : 2.3.0.0
GPS Latitude Ref       : South
GPS Longitude Ref      : East
GPS Altitude Ref       : Above Sea Level
Image Width            : 1429
Image Height           : 804
Encoding Process       : Baseline DCT, Huffman coding
Bits Per Sample        : 8
Color Components       : 3
Y Cb Cr Sub Sampling   : YCbCr4:2:0 (2 2)
Image Size             : 1429x804
Megapixels             : 1.1
GPS Altitude           : 134.3 m Above Sea Level
GPS Latitude           : 3 deg 13' 12.04" S
GPS Longitude          : 104 deg 39' 4.47" E
GPS Position           : 3 deg 13' 12.04" S, 104 deg 39' 4.47" E
zorin@master:~$
```

Gambar 19. Detail EXIF pada gambar "GIS\_image1.jpg"

```

File Size : 162 kB
File Modification Date/Time : 2023:11:27 14:49:42+07:00
File Access Date/Time : 2023:11:27 14:50:46+07:00
File Inode Change Date/Time : 2023:11:27 14:49:42+07:00
File Permissions : rw-rw-r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
Exif Byte Order : Big-endian (Motorola, MM)
GPS Version ID : 2.3.0.0
GPS Latitude Ref : South
GPS Longitude Ref : East
GPS Altitude Ref : Above Sea Level
Image Width : 1098
Image Height : 618
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 1098x618
Megapixels : 0.679
GPS Altitude : 122.2 m Above Sea Level
GPS Latitude : 3 deg 13' 8.67" S
GPS Longitude : 104 deg 39' 3.43" E
GPS Position : 3 deg 13' 8.67" S, 104 deg 39' 3.43" E
zorin@master:~$ █

```

Gambar 20. Detail EXIF pada gambar “GIS\_image2.jpg”

```

File Size : 214 kB
File Modification Date/Time : 2023:11:27 14:50:34+07:00
File Access Date/Time : 2023:11:27 14:50:42+07:00
File Inode Change Date/Time : 2023:11:27 14:50:34+07:00
File Permissions : rw-rw-r--
File Type : JPEG
File Type Extension : jpg
MIME Type : image/jpeg
Exif Byte Order : Big-endian (Motorola, MM)
GPS Version ID : 2.3.0.0
GPS Latitude Ref : South
GPS Longitude Ref : East
GPS Altitude Ref : Above Sea Level
Image Width : 1171
Image Height : 659
Encoding Process : Baseline DCT, Huffman coding
Bits Per Sample : 8
Color Components : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Image Size : 1171x659
Megapixels : 0.772
GPS Altitude : 134.2 m Above Sea Level
GPS Latitude : 3 deg 13' 12.26" S
GPS Longitude : 104 deg 39' 6.26" E
GPS Position : 3 deg 13' 12.26" S, 104 deg 39' 6.26" E
zorin@master:~$ █

```

Gambar 21. Detail EXIF pada gambar “GIS\_image3.jpg”

## Hasil Projek

Seperti yang telah diperlihatkan pada praktikum projek, output memperlihatkan gambar panorama dengan sisi bawah melengkung. Kualitas gambar tetap dijaga namun pemetaan pada panorama pertama (GIS\_image1) cukup berantakan. Hal ini disebabkan karena adanya perbedaan sudut saat mengambil gambar. Membuat program kebingungan untuk memindai area yang sama. Selain itu, panorama kedua (GIS\_image2) dan ketiga (GIS\_image3) tidak menstitching keseluruhan gambar. Alasan paling utamanya adalah saat pengambilan foto, gambar pada foto tidak presisi yang membuat foto tidak dapat diproses sistem.