Laporan Projek UAS
Georeferenced Image Stitching

Disusun Oleh :

Nama : Sanvic Dicaprio
NIM : 09011282227081
Kelas : SK5C

Dosen Pengampu :

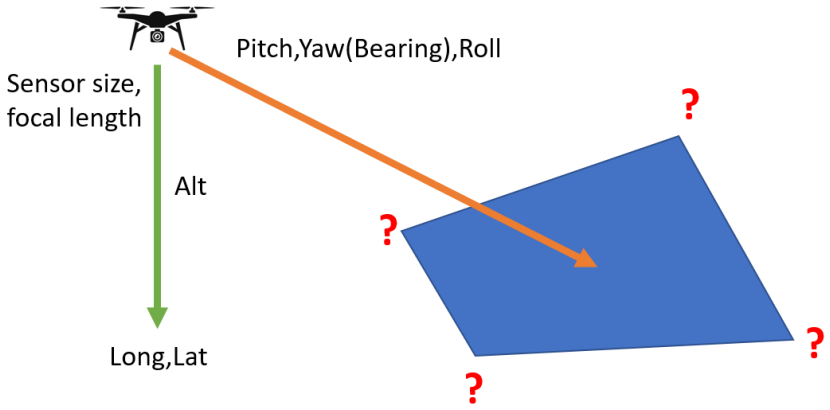
Adi Hermansyah, S.Kom., M.T.

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

Latar Belakang Projek

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 (*latitude*), 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 Projek

Tujuan pembuatan projek kali ini yaitu :

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

Metode Projek

Projek ini dibuat dengan menggunakan pemrograman python yang mengekstrak informasi GPS pada EXIF dengan library piexif. Pertama, program akan membaca informasi dari nilai latitude, longtitude, 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 (*Graptical User Interface*) dengan spesifikasi VM sebagai berikut :

- Jenis sistem operasi : Zorin OS 16.3 x86_64
- Prosessor : 12th Gen Intel i5-12500H 2 Core 3.110GHz
- RAM Virtual : 1024 MB
- Jaringan : Host-Only Adapter

```python
from mpi4py import MPI
import cv2
import argparse
import glob
import piexif
import numpy as np

cv2.ocl.setUseOpenCL(False)

def get_lat_lon_alt(exif):
    if 'GPS' not in exif:
        return None, None, None

    lat_tuple = exif['GPS'][piexif.GPSIFD.GPSLatitude]
    lon_tuple = exif['GPS'][piexif.GPSIFD.GPSLongitude]
    alt_tuple = exif['GPS'][piexif.GPSIFD.GPSAltitude]

    lat = convert_to_degrees(lat_tuple)
    lon = convert_to_degrees(lon_tuple)
    alt = alt_tuple[0] / alt_tuple[1]

    lat_ref = exif['GPS'][piexif.GPSIFD.GPSLatitudeRef]
    lon_ref = exif['GPS'][piexif.GPSIFD.GPSLongitudeRef]

    lat = lat * (-1 if lat_ref.upper() == 'S' else 1)
    lon = lon * (-1 if lon_ref.upper() == 'W' else 1)

    return lat, lon, alt

def convert_to_degrees(value):
    d, m, s = value
    return d[0] / d[1] + m[0] / (m[1] * 60) + s[0] / (s[1] * 3600)

def process_image(imagePath):
    image = cv2.imread(imagePath)
    exif_data = piexif.load(imagePath)
    lat, lon, alt = get_lat_lon_alt(exif_data)

    if lat is not None and lon is not None:
        return image, lat, lon, alt, exif_data
    else:
        return None

def load_images_with_gps_parallel(imagePaths):
    images_with_gps = []
    images_with_gps = []

    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()
    size = comm.Get_size()

    chunk_size = len(imagePaths) // size
    start_index = rank * chunk_size
    end_index = start_index + chunk_size if rank != size - 1 else len(imagePaths)

    local_image_paths = imagePaths[start_index:end_index]
    local_results = []

    for imagePath in local_image_paths:
        result = process_image(imagePath)
        if result is not None:
            local_results.append(result)

    all_results = comm.gather(local_results, root=0)

    if rank == 0:
        images_with_gps = [item for sublist in all_results for item in sublist]

    return images_with_gps

def stitch_images(images):
    num_images = len(images)

    if num_images < 2:
        print("[INFO] Insufficient images for stitching.")
```

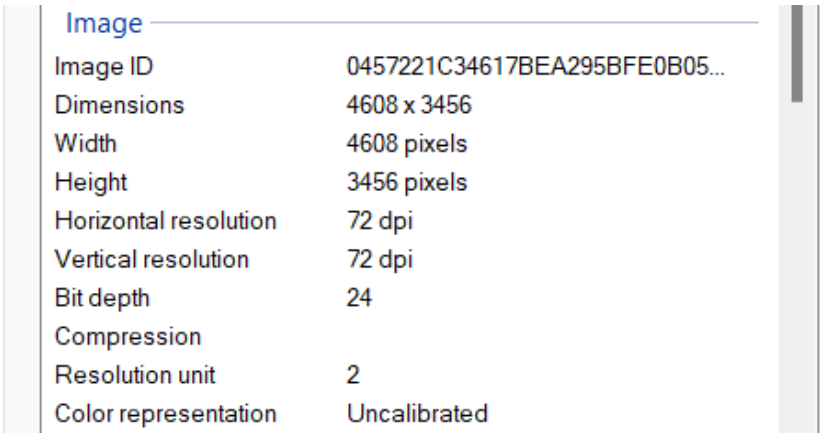
```
        print("[INFO] Insufficient images for stitching.")
        return None

    images_to_stitch = [img for img, _, _, _, _ in images]

    if cv2.__version__.startswith('3'):
        stitcher = cv2.createStitcher()
    else:
        stitcher = cv2.Stitcher_create()

    (status, stitched) = stitcher.stitch(images_to_stitch)

    if status == cv2.Stitcher_OK:

        aspect_ratio = images[0][0].shape[1] / images[0][0].shape[0]
        new_width = int(stitched.shape[0] * aspect_ratio)
        stitched_resized = cv2.resize(stitched, (new_width, stitched.shape[0]))

        return stitched_resized
    else:
        print(f"[INFO] Image stitching failed with status {status}")

        return None

def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--images", type=str, required=True,
        help="path to input directory of images to stitch")
    ap.add_argument("-o", "--output", type=str, required=True,
        help="path to the output image")
    args = vars(ap.parse_args())

    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()

    if rank == 0:
        print("[INFO] Memuat gambar...")

    imagePaths = glob.glob(args["images"] + "/*.jpg")
    images_with_gps = load_images_with_gps_parallel(imagePaths)

    if rank == 0:
        if not images_with_gps:
            print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
        else:
```

```
            print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
        else:
            images = [(img, lat, lon, alt, exif_data) for img, lat, lon, alt, exif_data in images_with_gps]

            print("[INFO] Merangkai gambar...")

            stitched_image = stitch_images(images)

            if stitched_image is not None:
                cv2.imwrite(args["output"], stitched_image)

                output_exif_data = piexif.load(args["output"])
                for _, _, _, _, exif_data in images:
                    if 'GPS' in exif_data:
                        output_exif_data['GPS'] = exif_data['GPS']
                exif_bytes = piexif.dump(output_exif_data)
                piexif.insert(exif_bytes, args["output"])

                print("[INFO] Penggabungan gambar berhasil.")
            else:
                print("[INFO] Penggabungan gambar gagal.")

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

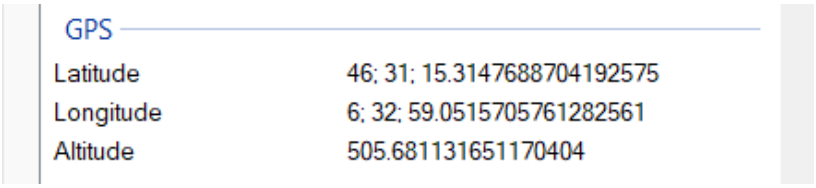
Gambar 2. Porgram python yang akan dijalankan dalam projek GIS

Gambar 3. VM yang akan dipakai dalam projek 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.



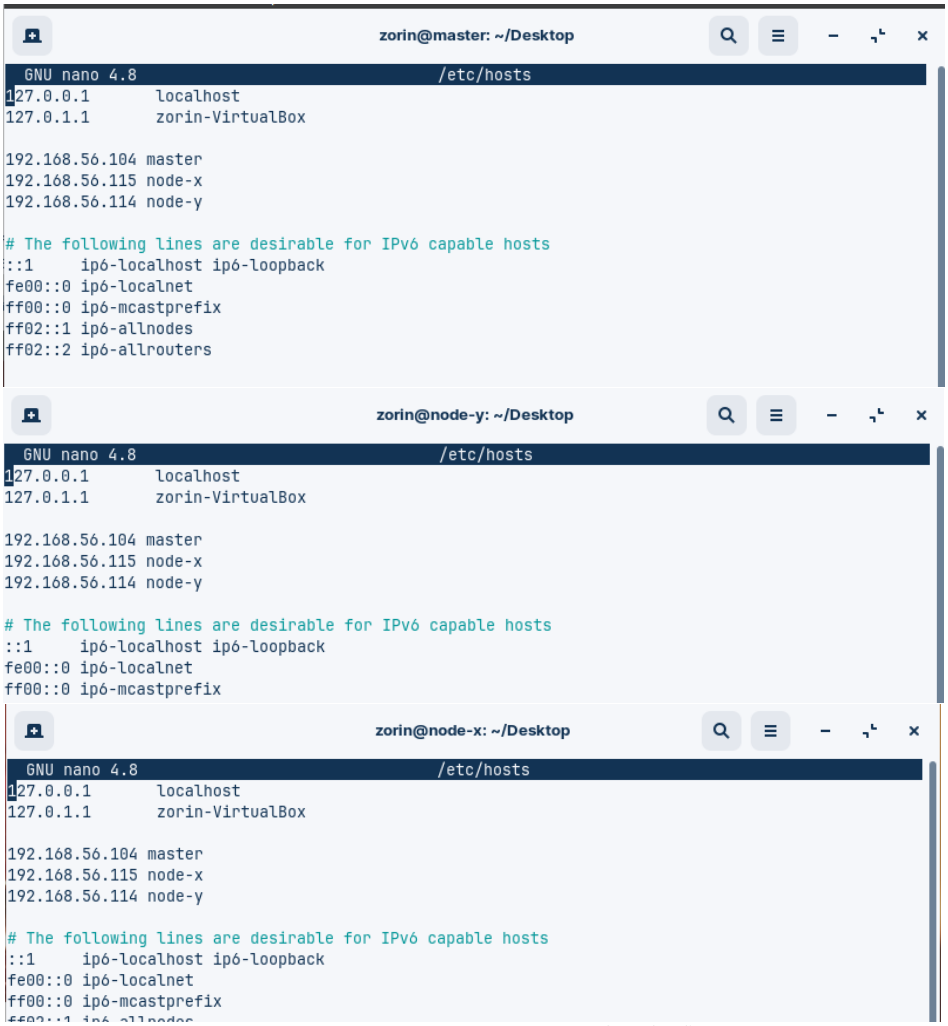
Gambar 4. Contoh detail fisik pada foto



Gambar 5. Contoh detail GPS pada foto

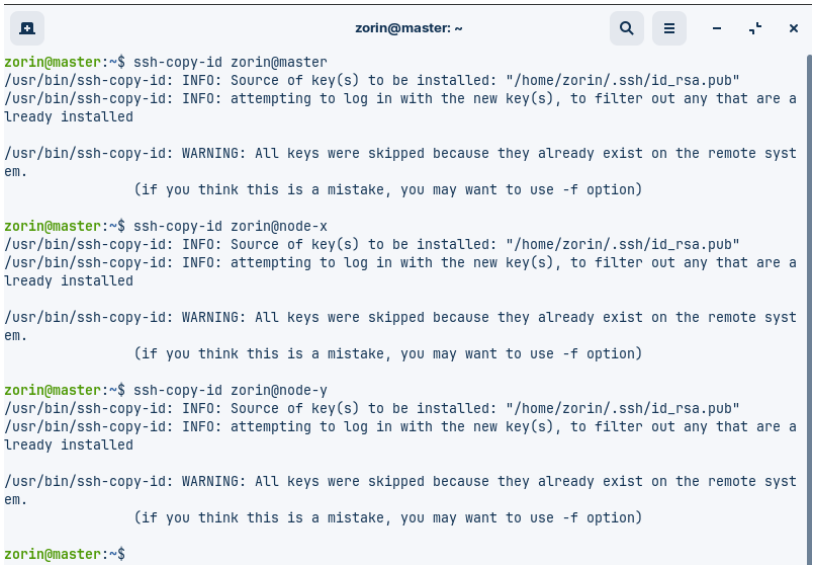
Praktikum Projek

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.



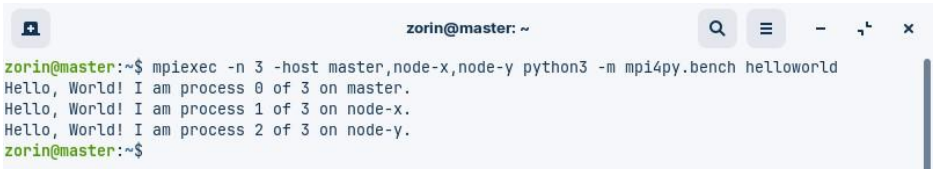
Gambar 6. Tampilan dari ketiga VM dalam file "/etc/hosts"

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>".

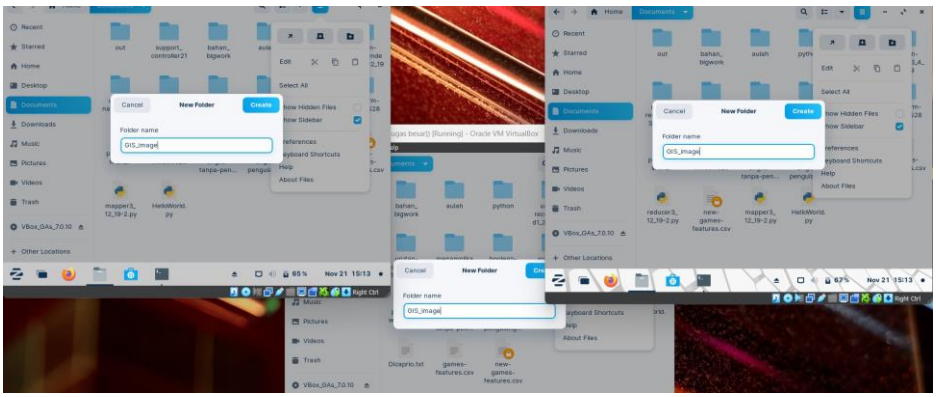


Gambar 7. Tampilan saat berhasil masuk sebagai localhost dan copy id ssh hostname di terminal

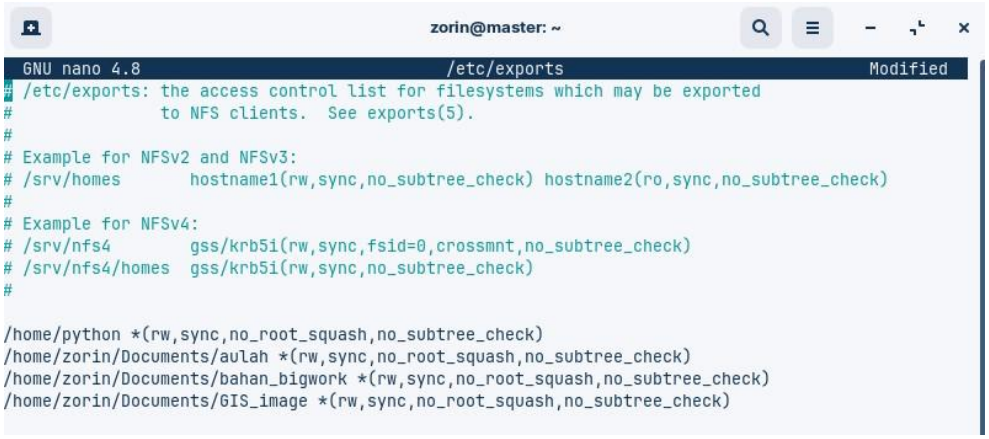
3. Cek apakah mpi telah berjalan secara paralel dengan mengetik perintah “mpiexec -n <jumlah_komputer> -host <nama_host1>,<nama_host2>,<nama_host3> python3 -m mpi4py.bench helloworld” (jumlah host tidak terlalu bergantung pada jumlah komputer. Kita dapat mengurangi jumlah komputer dibawah jumlah host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



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)”.

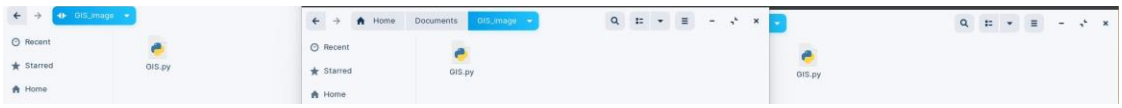


6. Simpan dan kembali. Ketik “sudo exportfs -a” untuk mengekspor ulang semua folder yang ingin dibagikan. Setelah itu, ketik “sudo systemctl restart nfs-kernel-server” untuk mereset sistem service nfs.

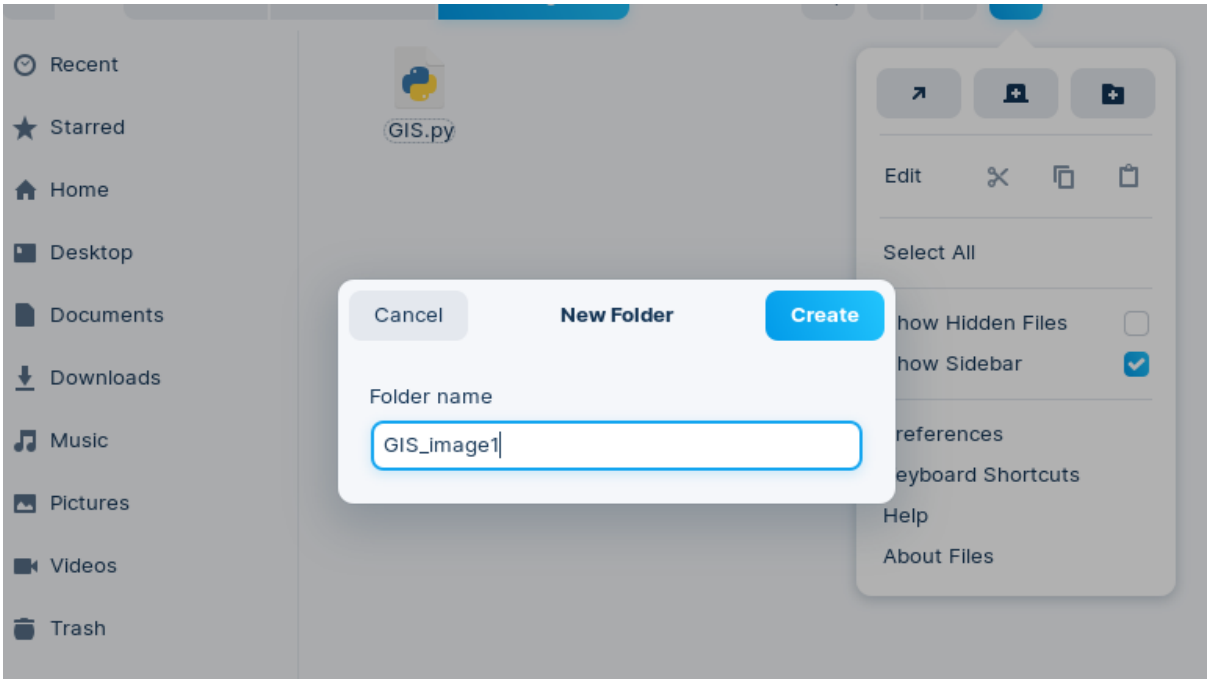
7. Kembali pada 2 VM lain. Ketik “sudo mount <Hostname_Master>:<lokasi folder di master> <lokasi folder di VM>”.

Gambar 8. Menghubungkan folder yang dibagikan master melalui layanan NFS

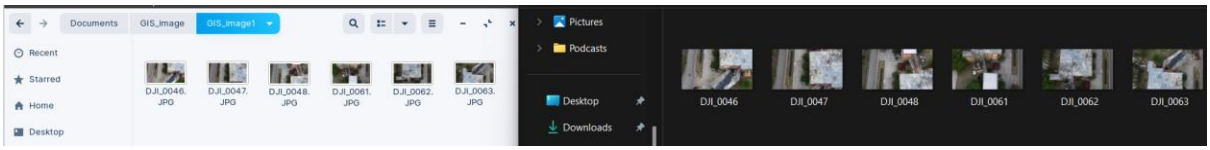
8. Cek dengan membuat file pada salah satu VM. Jika berhasil, maka akan tampil pada VM lain. (disini saya mengimpor codingan proyek GIS dengan nama GIS.py)



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

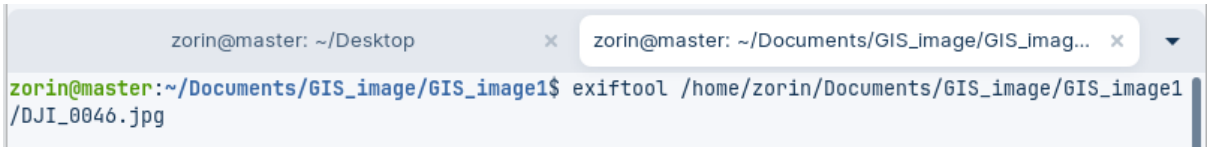


10. Ambil semua gambar dengan cara “drag and drop” semua gambar ke salah satu VM.

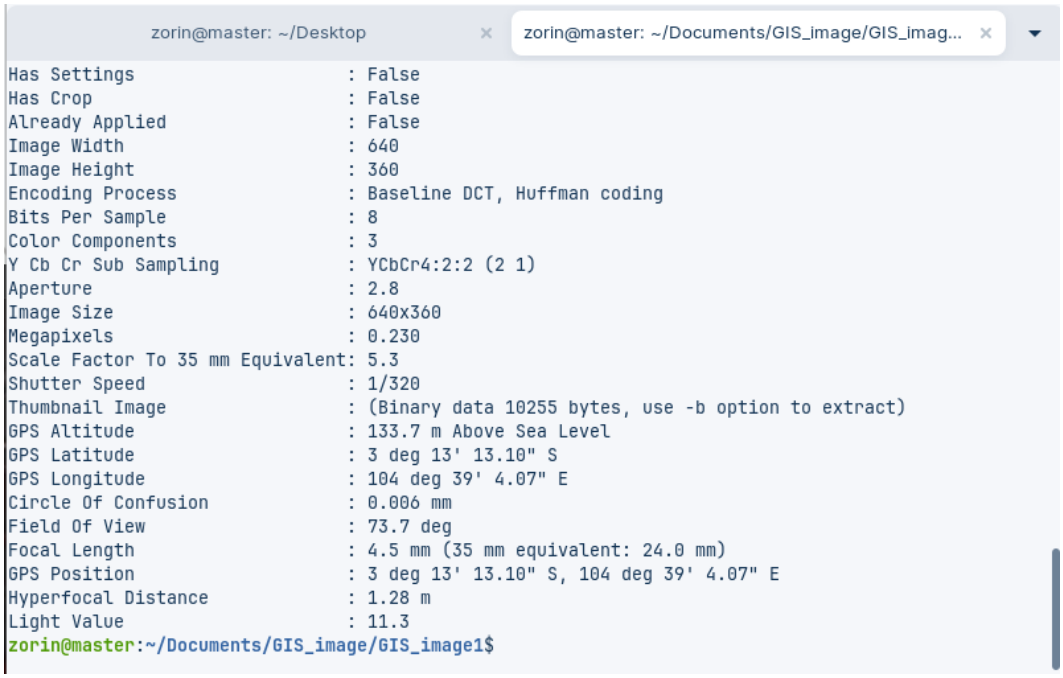


11. Samakan semua resolusi foto untuk memperkecil kemungkinan format yang error. Setelah itu, ubah format foto dari *.JPG menjadi *.jpg.

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

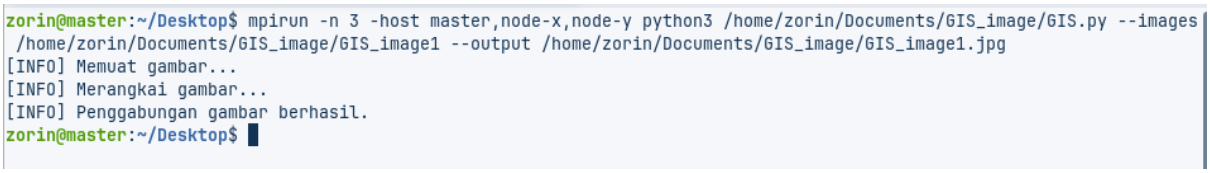


Gambar 13. Tool untuk mengetahui data EXIF

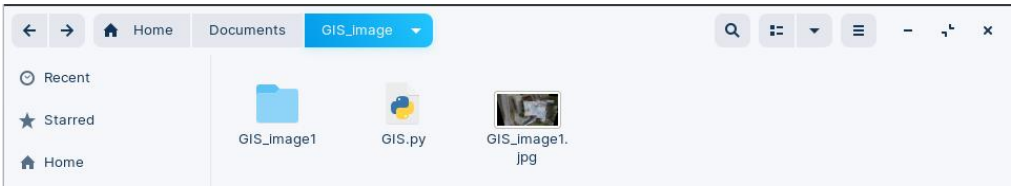


Gambar 14. Perintah exiftool setelah dieksekusi

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



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.