

Laporan Projek UAS Georeferenced Image Stitching



Disusun Oleh :

Nama : Sanvic Dicaprio
NIM : 09011282227081
Kelas : SK5C

Dosen Pengampu :

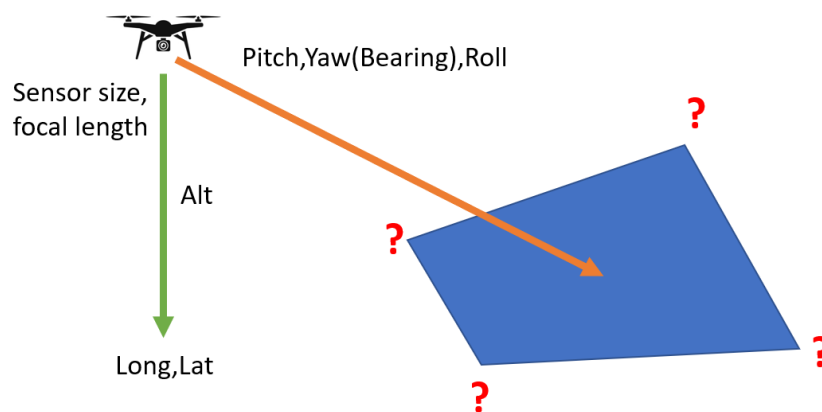
Adi Hermansyah, S.Kom., M.T.

**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. Program python akan 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 : 12th 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 plexif
6
7 def get_lat_lon_alt(exif):
8     if 'GPS' not in exif:
9         return None, None, None
10
11     lat_tuple = exif['GPS'][plexif.GPSIFD.GPSLatitude]
12     lon_tuple = exif['GPS'][plexif.GPSIFD.GPSLongitude]
13     alt_tuple = exif['GPS'][plexif.GPSIFD.GPSAltitude]
14
15     lat = convert_to_degrees(lat_tuple)
16     lon = convert_to_degrees(lon_tuple)
17     alt = alt_tuple[0] / alt_tuple[1]
18
19     lat_ref = exif['GPS'][plexif.GPSIFD.GPSLatitudeRef]
20     lon_ref = exif['GPS'][plexif.GPSIFD.GPSLongitudeRef]
21
22     lat = lat * (-1 if lat_ref.upper() == 'S' else 1)
23     lon = lon * (-1 if lon_ref.upper() == 'W' else 1)
24
25     return lat, lon, alt
26
27 def convert_to_degrees(value):
28     d, m, s = value
29     return d[0] / d[1] + m[0] / (m[1] * 60) + s[0] / (s[1] * 3600)
30
31 def process_image(imagePath):
32     image = cv2.imread(imagePath)
33     exif_data = plexif.load(imagePath)
34     lat, lon, alt = get_lat_lon_alt(exif_data)
35
36     if lat is not None and lon is not None:
37         return image, lat, lon, alt
38     else:
39         return None
40
41 def load_images_with_gps_parallel(imagePaths):
42     images_with_gps = []
43
44     comm = MPI.COMM_WORLD
45     rank = comm.Get_rank()
46     size = comm.Get_size()
47
48     chunk_size = len(imagePaths) // size
49     start_index = rank * chunk_size
50     end_index = start_index + chunk_size if rank != size - 1 else len(imagePaths)
51
52     local_image_paths = imagePaths[start_index:end_index]
53     local_results = []
54
55     for imagePath in local_image_paths:
56         result = process_image(imagePath)
57         if result is not None:
58             local_results.append(result)
59
60     all_results = comm.gather(local_results, root=0)
61
62     if rank == 0:
63         images_with_gps = [item for sublist in all_results for item in sublist]
64
65     return images_with_gps
66
67 def main():
68     ap = argparse.ArgumentParser()
69     ap.add_argument("-i", "--images", type=str, required=True,
70                     help="path to input directory of images to stitch")
71     ap.add_argument("-o", "--output", type=str, required=True,
72                     help="path to the output image")
73     args = vars(ap.parse_args())
74
75     comm = MPI.COMM_WORLD
76     rank = comm.Get_rank()
77
78     if rank == 0:
79         print("[INFO] Memuat gambar...")
80
81     imagePaths = glob.glob(args["images"] + "/*.jpg")
82     images_with_gps = load_images_with_gps_parallel(imagePaths)
83
84     if rank == 0:
85         if not images_with_gps:
86             print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
87
88     print("[INFO] Tidak ada data GPS dalam metadata EXIF.")
89     else:
90         images = [img for img, _, _, _ in images_with_gps]
91
92     print("[INFO] Merangkai gambar...")
93
94     if cv2.__version__.startswith('3'):
95         stitcher = cv2.createStitcher()
96     else:
97         stitcher = cv2.Stitcher_create()
98
99     (status, stitched) = stitcher.stitch(images)
100
101     if status == 0:
102         cv2.imwrite(args["output"], stitched)
103         cv2.imshow("Stitched", stitched)
104         cv2.waitKey(0)
105
106     else:
107         print("[INFO] Penggabungan gambar gagal ({}).format(status))
108
109 if __name__ == "__main__":
110     main()

```

Gambar 2. Program python yang akan dijalankan dalam proyek GIS



Gambar 3. VM yang akan dipakai dalam proyek GIS

Untuk objek panorama, kami mengambil 50 foto dari luar yang menampilkan informasi GPS pada tahun 2016. 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

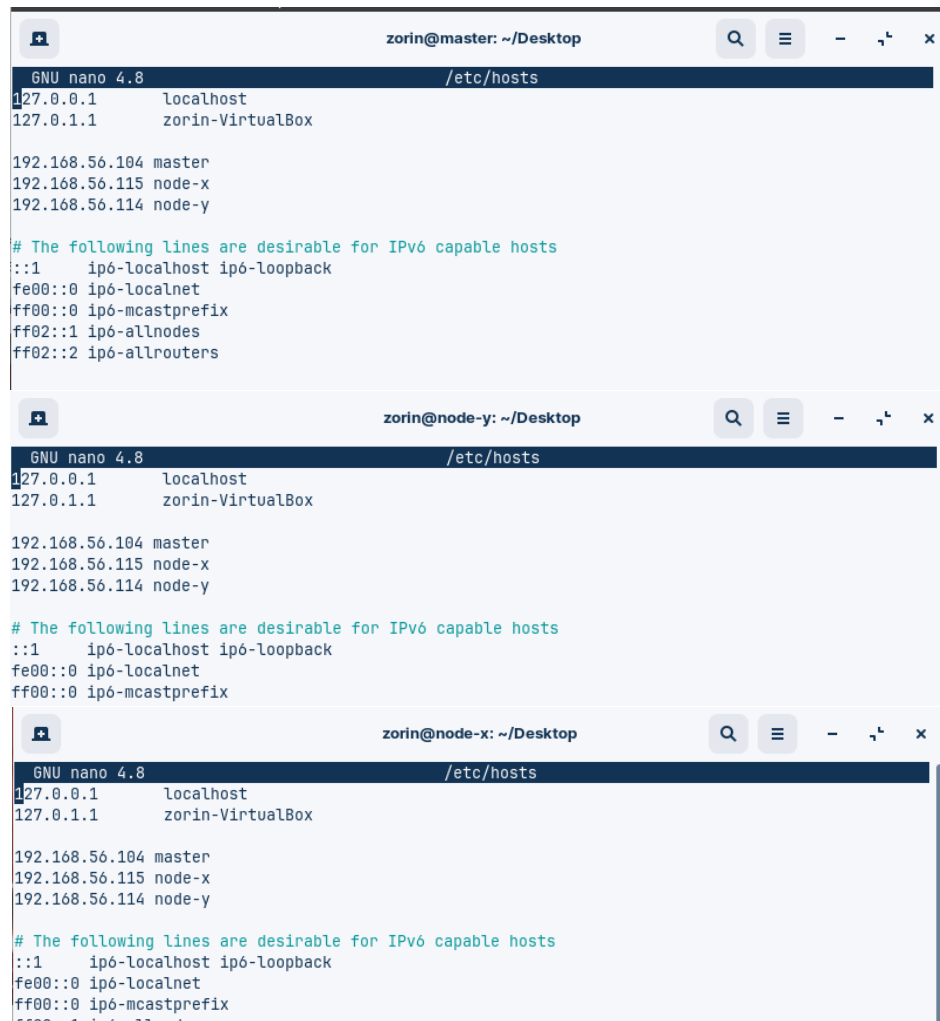
Gambar 4. Detail fisik pada foto

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

Gambar 5. Detail GPS pada foto

Praktikum dan Hasil 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>`”.



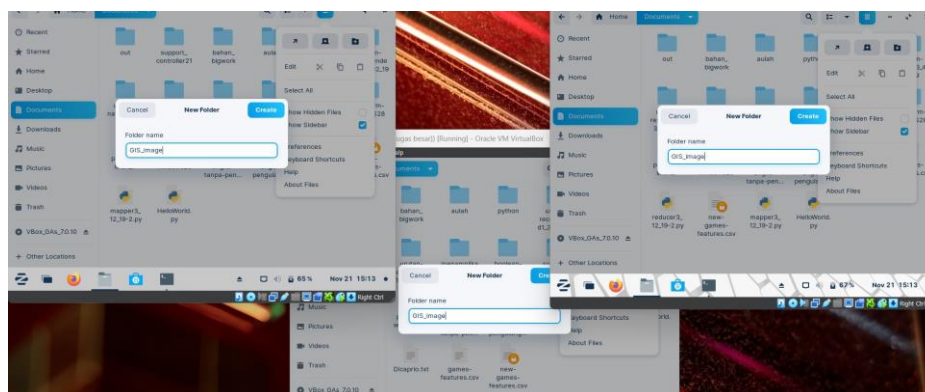
```
zorin@master: ~  
zorin@master:~$ ssh-copy-id zorin@master  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/zorin/.ssh/id_rsa.pub"  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
  
/usr/bin/ssh-copy-id: 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>”.

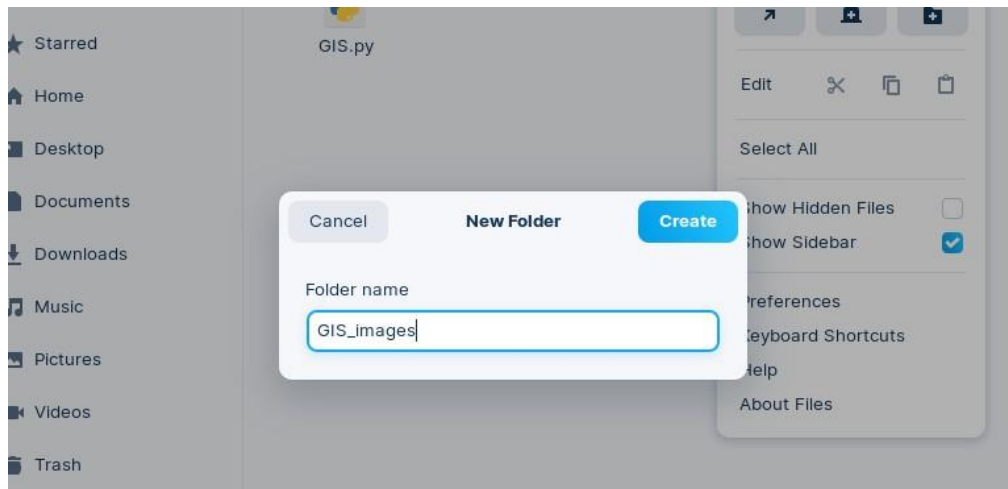
```
zorin@node-y: ~/Desktop  
zorin@node-y:~/Desktop$ sudo mount master:/home/zorin/Documents/GIS_image /home/zorin/Documents/GIS_image/  
zorin@node-y:~/Desktop$  
  
zorin@node-x: ~/Desktop  
zorin@node-x:~/Desktop$ sudo mount master:/home/zorin/Documents/GIS_image /home/zorin/Documents/GIS_image/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

Gambar 8. Menghubungkan folder yang dibagikan master melalui layanan NFS

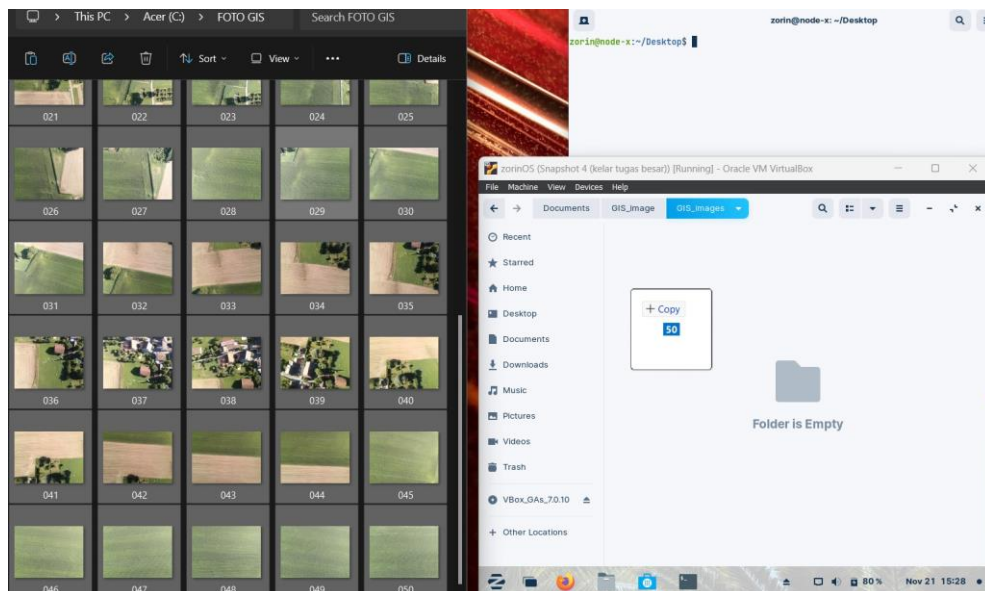
8. Cek dengan membuat file pada salah satu VM. Jika berhasil, maka akan tampil pada VM lain. (disini saya mengimpor codingan proyek GIS dengan nama GIS.py)



9. Buat folder yang akan menampung gambar proyek GIS.



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



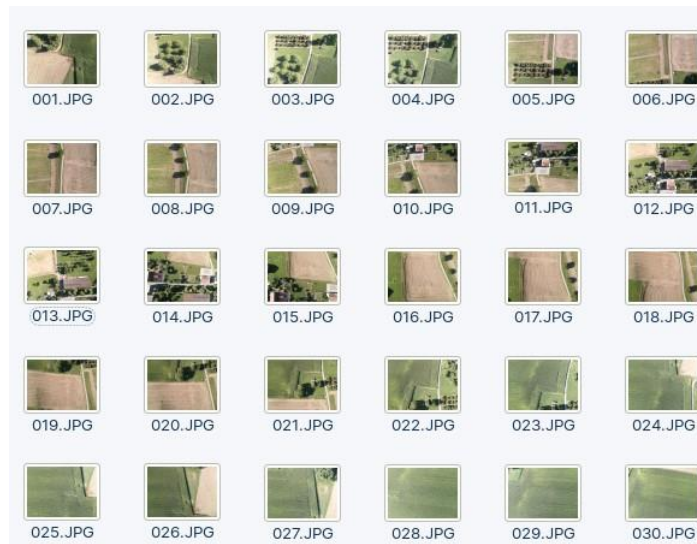
11. Samakan semua resolusi foto untuk memperkecil kemungkinan format yang error.



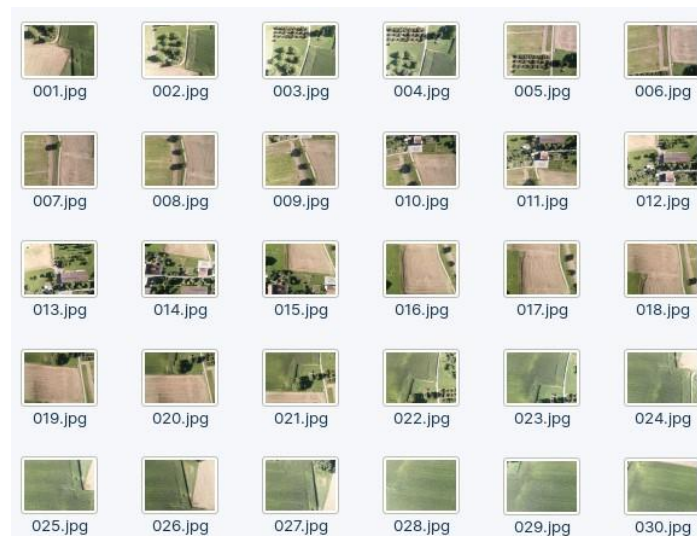
12. Pastikan untuk mengubahnya menjadi *.jpg. Seringkali format *.jpg saat di pindahkan dalam linux menjadi *.JPG. Hal ini mengakibatkan program tak dapat membaca EXIF pada gambar.


```
zorin@master: ~/Documents/GIS_image/GIS_images
zorin@master:~/Documents/GIS_image/GIS_images$ mpirun -n 3 -host master,node-x,node-y python3 /home/zorin/Documents/GIS_image/GIS.py --images /home/zorin/Documents/GIS_image/GIS_images --output /home/zorin/Documents/GIS_image/GIS.jpg
[INFO] Memuat gambar...
[INFO] Tidak ada data GPS dalam metadata EXIF.
zorin@master:~/Documents/GIS_image/GIS_images$
```

Gambar 9. Data EXIF tidak terbaca jika format gambar tidak diubah ke *.jpg



Gambar 10. Format gambar sebelum diubah



Gambar 11. Format gambar setelah diubah

```
zorin@master: ~/Documents/GIS_image/GIS_images
zorin@master:~/Documents/GIS_image/GIS_images$ mogrify -resize 1024x768 *.JPG
zorin@master:~/Documents/GIS_image/GIS_images$
```

Gambar 12. Proses mengubah format dari *.JPG ke *.jpg

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

```
zorin@master: ~/Documents/GIS_image/GIS_images
zorin@master:~/Documents/GIS_image/GIS_images$ exiftool 001.jpg
```

Gambar 13. Tool untuk mengetahui data EXIF

```
zorin@master: ~/Documents/GIS_image/GIS_images
+20A6vsEfH70BAAAA0ZtgCwAAADVbK8+ISAUViVgDb0BAAAAIZ9gCwAAAAABmnZo+vtY6vgzQEb0BAAACaNgCwAAAAByISU+6
ZzuvaIPi70BAAAA8aZgCwAAAAAvzpw+7Y0YvmbDjLwBAAAA2agCwAAAAAIyJY+7c50vtjB07wBAAAAwa5gCwAAAAATpE+VfN
Fvj7wCL0BAAAAy7JgCwAAAAA2SJw+6eLPvn6+jLwBAAAA87ZgCwAAAAADnbKk+YmLfvsL9dbwBAAAAm7pgCwAAAAA0c68+kd86v
Deg7wBAAAAg75gCwAAAAADtr6Y+SKBhvhiAC7wBAAAAa8JgCwAAAAACFI48+iie0vvC807wBAAAA.
Image Width           : 1024
Image Height          : 768
Encoding Process      : Baseline DCT, Huffman coding
Bits Per Sample       : 8
Color Components      : 3
Y Cb Cr Sub Sampling : YCbCr4:2:0 (2 2)
Aperture              : 2.3
Image Size            : 1024x768
Megapixels            : 0.786
Scale Factor To 35 mm Equivalent: 5.7
Shutter Speed         : 1/707
Create Date           : 1970:01:01 00:03:10.903393
Date/Time Original    : 2016:07:29 07:13:49.391860
Modify Date           : 2016:07:29 07:13:49.391860
GPS Altitude          : 505.6 m Above Sea Level
GPS Date/Time         : 2016:07:29 07:13:48.18699828Z
GPS Latitude          : 46 deg 31' 15.31" N
GPS Longitude         : 6 deg 32' 59.05" E
Circle Of Confusion   : 0.005 mm
Field Of View         : 65.5 deg
Focal Length          : 4.9 mm (35 mm equivalent: 28.0 mm)
GPS Position          : 46 deg 31' 15.31" N, 6 deg 32' 59.05" E
Hyperfocal Distance   : 1.98 m
Light Value           : 11.9
zorin@master:~/Documents/GIS_image/GIS_images$
```

Gambar 14. Perintah exiftool setelah dieksekusi

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

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

15. Ketika selesai, cek pada direktori yang telah ditentukan output.



Gambar 15. Perintah MPI setelah dieksekusi



Gambar 16. Hasil gambar dari "GIS.jpg"

Perbandingan GIS dan Non-GIS

Setelah melakukan praktikum proyek diatas. Kami akan membandingkan panorama dari 25 foto yang sama antara menggunakan metode GIS dan metode Non-GIS. Hal ini bertujuan untuk mengetahui perbedaan secara langsung antara panorama biasa dan panorama GIS. Program Non-GIS merupakan program python yang telah dipakai sebelumnya pada laporan 5 yaitu "image stitching.". Program akan dijalankan pada 1 VM untuk menghilangkan dugaan terkait output yang berbeda antara pemrosesan paralel atau serial.


```

1 from imutils import paths
2 import numpy as np
3 import argparse
4 import imutils
5 import cv2
6
7 ap = argparse.ArgumentParser()
8 ap.add_argument("-i", "--images", type=str, required=True,
9                 help="path to input directory of images to stitch")
10 ap.add_argument("-o", "--output", type=str, required=True,
11                help="path to the output image")
12 args = vars(ap.parse_args())
13
14 print("[INFO] loading images...")
15 imagePath = sorted(list(paths.list_images(args["images"])))
16 images = []
17
18 for imagePath in imagePath:
19     image = cv2.imread(imagePath)
20     images.append(image)
21
22 print("[INFO] stitching images...")
23 stitcher = cv2.createStitcher() if imutils.is_cv3() else cv2.Stitcher_create()
24 (status, stitched) = stitcher.stitch(images)
25
26 if status == 0:
27     # write the output stitched image to disk
28     cv2.imwrite(args["output"], stitched)
29
30     # display the output stitched image to our screen
31     cv2.imshow("Stitched", stitched)
32     cv2.waitKey(0)
33
34 else:
35     print("[INFO] image stitching failed ({}).format(status))

```

Gambar 17. Isi program python dari Non-GIS

Berikut adalah 2 panorama yang menggunakan GIS dan Non-GIS :



Gambar 18. Panorama dengan metode GIS



Gambar 19. Panorama dengan metode Non-GIS

Menurut kami, cukup sulit membedakan keduanya apabila dilihat sekilas. Hal itu karena panorama baik GIS atau Non-GIS mempunyai ciri khas yang tidak menonjol. Apalagi jika kedua panorama ini saling menggantikan atau menutupi. Namun setelah dilihat secara seksama dan memperbesar gambar, ada beberapa momen dimana GIS terlihat lebih unggul dibandingkan dengan Non-GIS. Begitu juga sebaliknya. Berikut beberapa perbedaan dari hasil panorama yang telah diperlihatkan :

1. Jalan



Gambar 20. Perbandingan 1 GIS (Kiri) dan Non-GIS (kanan)

Terlihat pada gambar tersebut, program Non-GIS mengira bahwa garis jalan pertama dan garis jalan selanjutnya tidak terhubung/terputus. Membuat program tersebut harus memaksa *stitching* agar terlihat menyatu dan mengakibatkan visual jalanan yang seolah terlihat seperti “mengunung” (naik kemudian turun). Sementara pada GIS, program telah membaca informasi GPS pada gambar tersebut. Sehingga program dapat memperkirakan berapa ketinggian pengambilan foto dan sudut pengambilan. Membuat jalanan yang “mungkin” terlihat miring dapat dibuat lurus kembali. Tapi sebagai kompensasi, kondisi garis tanah di sebelah kiri pohon GIS sama seperti jalan di panorama Non-GIS.

2. Bangunan persegi panjang dekat mobil



Gambar 21. Perbandingan 2 GIS (Kiri) dan Non-GIS (kanan)

Pada gambar ini, detail bangunan dekat pohon pada Non-GIS terlihat berada disebelah utara pohon. Sementara pada GIS, bangunan seolah berada dibawah pohon. Hal yang mungkin terjadi adalah karena pada saat pengambilan gambar, drone tak sengaja berada terlalu miring pada pohon dibandingkan rumah. Memanipulasi sudut kemiringan bujur dan lintang dalam pengambilan dan pemrosesan panorama GIS. Namun Non-GIS

memprosesnya seperti 2 bangunan yang berbeda. Sehingga membuat 2 perspektif pada gambar terpecah saat pembuatan panorama.

3. Detail kecil pada tempat tertentu



Gambar 22. Perbandingan 3 GIS (Kiri) dan Non-GIS (kanan)



Gambar 23. Perbandingan 4 GIS (Kiri) dan Non-GIS (kanan)

Detail seperti pagar tanaman (bawah) dalam panorama Non-GIS tetap ditampilkan, sementara pada GIS dihilangkan. Lalu karena point 1 yang membahas detail jalanan pada GIS dioptimalkan berdasarkan perhitungan, ia juga mengorbankan tingkat presisi pada garis batas jalan. Membuat jalan tersebut terlihat patah / miring.

Kesimpulan

GIS (*Georeferenced Image Stitching*) adalah proses penggabungan gambar membentuk panorama yang memakai nilai GPS dalam menentukan hasil akhir. Oleh karena itu, GIS lebih sering dipakai pada bidang yang sering berhubungan dengan data geografi. GIS juga membutuhkan tingkat presisi yang bagus pada alat bantu seperti drone agar tetap stabil dalam pengambilan objek. Perbedaan paling mendasar GIS dan Non-GIS adalah Non-GIS lebih memprioritaskan visual dengan efek panorama yang lebih enak dilihat dalam 1 arah. Sementara GIS lebih memprioritaskan hasil akhir yang sesuai dengan perhitungan masing-masing latitude, longitude, dan altitude pada semua gambar. Sehingga beberapa detail

yang kurang banyak muncul dalam foto akan dihilangkan agar panorama GIS terlihat lebih mirip dengan lokasi aslinya. Makan oreo bersama kekasih, saya dicaprio dan terimakasih. Sfx*
teng teng teng teng teng teng teng...