Perancangan dan Pengembangan Sistem Alat Pemindai Sedimen dengan Implementasi Image Stitching Berbasis Python-OpenCV, Flask, Supabase, dan Protokol MQTT

1st Bayu Lesmana
School of Electrical Engineering
Telkom University
Bandung, Indonesia
bayuish@student.telkomuniversity.ac.id

2nd Dini Fitria Arifah
School of Electrical Engineering
Telkom University
Bandung, Indonesia
dinifitri@student.telkomuniversity.ac.id

3rd Muhamad Auli'a Ardani
School of Electrical Engineering
Telkom University
Bandung, Indonesia
muhamadardani@student.telkomuniversity.ac.id

***Abstrak* — Dokumentasi visual sedimen merupakan bagian penting dalam studi geologi dan pemetaan lingkungan. Metode manual yang mengandalkan pengambilan gambar terpisah sering menghadapi kendala inkonsistensi posisi, pencahayaan, dan sudut pandang, yang berakibat pada penurunan kualitas data. Selain itu, penggunaan perangkat konvensional seperti pemindai laboratorium beresolusi tinggi tidak efisien karena biaya dan pemeliharaan yang tinggi. Untuk mengatasi permasalahan ini, dikembangkan sistem pemindai sedimen berbasis image stitching yang memanfaatkan Python, OpenCV, dan Flask untuk pengolahan citra secara otomatis. Protokol MQTT digunakan untuk komunikasi data real-time antara perangkat pemindai dan server, sedangkan Supabase dimanfaatkan untuk penyimpanan dan pengelolaan data citra secara terpusat di cloud. Pengujian dilakukan dengan 3 size sedimen yakni 30 cm, 50 cm dan 100 cm menunjukkan keberhasilan transmisi data mencapai 96,2% dengan latensi rata-rata 1,12 detik. Proses image stitching menghasilkan citra utuh dalam waktu kurang dari 10 detik untuk berbagai lintasan. Sistem ini terbukti efektif dalam meningkatkan konsistensi, efisiensi, dan keandalan dokumentasi visual sedimen.**

Kata kunci — Image stitching, Python, OpenCV, Flask, MQTT, Supabase

I. PENDAHULUAN

Dokumentasi visual sedimen memegang peranan penting dalam studi geologi, pemetaan lingkungan, dan konservasi sumber daya alam [1]. Data visual yang presisi sangat diperlukan untuk mengidentifikasi karakteristik fisik sedimen dan mendukung analisis ilmiah lanjutan. Namun, metode dokumentasi konvensional sering kali tidak konsisten akibat variasi pencahayaan, sudut pengambilan gambar, dan perbedaan resolusi. Di sisi lain, alat laboratorium seperti Multi-Sensor Core Logger (MSCL-S) memang mampu menghasilkan citra berkualitas tinggi, tetapi tidak efisien untuk digunakan di lapangan karena harga yang tinggi dan kebutuhan pemeliharaan yang kompleks [2]. Menghadapi keterbatasan tersebut, dikembangkan sistem pemindai sedimen berbasis image stitching untuk menghasilkan citra utuh dari fragmen-fragmen gambar yang diambil secara

terpisah. Sistem ini memanfaatkan Python dengan pustaka OpenCV sebagai inti proses penggabungan citra, sementara Flask digunakan sebagai server untuk mengelola alur pemrosesan. Data dikirimkan secara real-time menggunakan protokol MQTT, memastikan komunikasi yang efisien dan andal. Seluruh citra hasil penggabungan beserta metadata disimpan di Supabase sebagai basis data cloud, yang memungkinkan akses cepat, terdistribusi, dan aman.

Pengembangan sistem ini diharapkan mampu meningkatkan efisiensi dan konsistensi dokumentasi visual sedimen, mengurangi ketergantungan pada metode manual, serta memberikan alternatif yang lebih ekonomis dan fleksibel dibandingkan perangkat konvensional.

II. KAJIAN TEORI

Kajian teori berfungsi untuk memberikan landasan konseptual terhadap teknologi-teknologi yang digunakan dalam penelitian ini. Pemahaman yang baik terhadap teori dasar ini menjadi kunci dalam merancang sistem pemindai sedimen berbasis image stitching yang efisien. Adapun teknologi yang menjadi fokus kajian meliputi Python, Flask, OpenCV, MQTT, dan Supabase

A. Image Stiching

Image stitching adalah teknik penggabungan beberapa citra menjadi satu panorama utuh beresolusi tinggi [3]. Teknik ini krusial untuk memperluas bidang pandang dokumentasi sedimen tanpa kehilangan detail spasial, sehingga seluruh sampel dapat terekam dalam satu citra menyeluruh.

Pada sistem SedimTrack, proses stitching dilakukan otomatis menggunakan Python sebagai bahasa utama, Flask sebagai pengatur alur backend, dan OpenCV sebagai pustaka pemrosesan citra.

Alur kerja sistem sebagai berikut:

1. Mobile App mengambil serangkaian fragmen citra sedimen secara berurutan sepanjang lintasan pemindaian, lalu mengunggahnya ke Supabase.
2. PC Pemrosesan memantau status unggahan melalui MQTT. Setelah menerima sinyal bahwa semua citra telah terunggah, PC mengunduh seluruh fragmen dari Supabase.
3. OpenCV melakukan proses stitching yang meliputi aligning, cropping, dan rotasi korektif berdasarkan urutan pengambilan. Pendekatan ini memanfaatkan stabilitas posisi kamera sehingga tidak memerlukan pencocokan fitur kompleks, membuat proses lebih cepat dan efisien.
4. Hasil panorama diunggah kembali ke Supabase dan tersedia untuk diakses melalui Mobile App pada menu hasil.

Metode ini terbukti efisien karena proses dapat diselesaikan dalam waktu kurang dari 10 detik per lintasan, dengan kualitas hasil yang konsisten berkat kontrol posisi kamera yang presisi.

B. Pyhton

Python merupakan bahasa pemrograman tingkat tinggi yang bersifat interpreted, open source, dan multi-platform [4]. Bahasa ini terkenal karena sintaksnya yang sederhana serta dukungan pustaka yang luas untuk berbagai kebutuhan seperti komputasi ilmiah, pengolahan data, machine learning, hingga pengolahan citra.

Keunggulan Python antara lain:

1. Sintaks sederhana yang memudahkan pengembangan cepat (rapid development).
2. Ekosistem pustaka luas seperti NumPy, Pandas, dan OpenCV yang mendukung pengolahan data dan citra.
3. Portabilitas tinggi, dapat berjalan di berbagai sistem operasi tanpa banyak penyesuaian kode.

Dalam penelitian ini, Python digunakan sebagai bahasa utama pada sisi server untuk mengelola proses pemrosesan citra sedimen. Proses image stitching memanfaatkan pustaka OpenCV yang terintegrasi dalam ekosistem Python, sehingga memberikan fleksibilitas dan efisiensi dalam pengembangan.

C. Flask

Flask adalah microframework web berbasis Python yang fleksibel dan ringan [5]. Dalam sistem SedimTrack, Flask berfungsi sebagai pengelola alur backend antara pengguna, proses pengolahan citra, dan modul komunikasi berbasis MQTT. Peran Flask dalam sistem ini anatara lain :

1. Menerima notifikasi dari MQTT bahwa seluruh gambar hasil pemindaian telah diunggah ke Supabase.
2. Mengunduh seluruh gambar dari Supabase sesuai urutan pemindaian.
3. Menjalankan proses image stitching otomatis menggunakan OpenCV (stitching, cropping, rotasi).
4. Mengunggah hasil stitching kembali ke Supabase pada direktori */Image/Results*.
5. Menyediakan API bagi aplikasi mobile untuk mengakses dan menampilkan hasil stitching pada menu *Hasil*.

Flask juga memungkinkan pengendalian stitching secara interaktif melalui antarmuka web atau API yang dapat dipicu oleh sistem otomatis berdasarkan status pemindaian.

D. MQTT sebagai Protokol Komunikasi Antar Perangkat.

Message Queuing Telemetry Transport (MQTT) merupakan protokol komunikasi ringan (lightweight) yang menggunakan arsitektur publish-subscribe [6]. MQTT dirancang untuk aplikasi Internet of Things (IoT) dan machine-to-machine (M2M) yang membutuhkan efisiensi pada jaringan dengan bandwidth terbatas, latensi rendah, dan reliabilitas tinggi. MQTT beroperasi pada lapisan aplikasi model TCP/IP. Umumnya, koneksi dilakukan melalui:

- Port TCP 1883 untuk koneksi non-terenkripsi.
- Port TCP 8883 untuk koneksi terenkripsi dengan Transport Layer Security (TLS).

Implementasi TLS menjadi sangat penting pada infrastruktur publik untuk menjamin keamanan, integritas, dan kerahasiaan data yang dikirimkan melalui jaringan [7]. Dalam sistem SedimTrack, MQTT digunakan sebagai mekanisme sinkronisasi status antar perangkat yang terhubung, yaitu aplikasi pengendali, perangkat pemindai, dan komputer pemroses stitching.

1. Arsitektur Publish-Subscribe dalam SedimTrack
   - Publisher (Aplikasi Mobile)

   Aplikasi mobile mempublikasikan pesan ke broker MQTT pada topik tertentu, misalnya scanner/status, dengan payload berupa flag (contohnya nilai boolean true) yang menandakan bahwa seluruh gambar telah berhasil diunggah ke Supabase.
   - Broker MQTT

   Broker bertindak sebagai perantara yang mendistribusikan pesan dari publisher ke subscriber sesuai topik yang sama.
   - Subscriber (Komputer Pemroses)

   Komputer pemroses yang berlangganan (subscribe) pada topik scanner/status akan menerima pesan dan secara otomatis memulai proses image stitching menggunakan Python, Flask, dan OpenCV.

2. Quality of Service (QoS) dan Retained Message

Untuk menjaga keandalan pengiriman pesan, sistem menggunakan QoS Level 1 yang menjamin pesan terkirim setidaknya satu kali. Meskipun terdapat kemungkinan duplikasi pesan, QoS 1 memberikan keseimbangan antara keandalan transmisi dan efisiensi bandwidth. Selain itu, digunakan fitur Retained Message. Dengan retained message, broker menyimpan salinan pesan terakhir pada topik yang bersangkutan, sehingga subscriber baru yang berlangganan akan langsung menerima pesan terbaru tanpa harus menunggu publisher mengirimkan ulang.

3. Keunggulan MQTT pada Sistem SedimTrack

- Ringan dan efisien, cocok untuk komunikasi status real-time.
- Andal, dengan QoS 1 yang menjamin pengiriman pesan penting minimal satu kali.
- Reaktif, dengan arsitektur event-driven yang memungkinkan sinkronisasi otomatis antar perangkat.

Dengan kombinasi arsitektur publish-subscribe, QoS 1, dan retained message, MQTT terbukti andal untuk mengatur alur kerja real-time antara proses akuisisi gambar, pemrosesan stitching, dan penyimpanan hasil pada Supabase.

E. Open CV

OpenCV (Open Source Computer Vision Library) merupakan pustaka fundamental dalam bidang computer vision dan pemrosesan citra digital [8]. Dalam sistem SedimTrack, OpenCV menjadi inti proses image stitching, yang menggabungkan beberapa citra hasil pemindaian menjadi citra panorama. OpenCV dipilih karena:

- Efisiensi komputasi: mendukung manipulasi citra (resize, cropping, masking, rotasi, transformasi geometris) langsung di Python.
- Fleksibilitas metode: mampu mengakomodasi baik stitching berbasis pencocokan fitur maupun metode deterministik.

Pada sistem SedimTrack, digunakan pendekatan deterministik yang memanfaatkan kestabilan posisi kamera yang dikontrol oleh sistem motorik presisi. Pendekatan ini berbeda dari metode berbasis deteksi fitur seperti SIFT atau ORB, karena:

- Lebih ringan secara komputasi.
- Lebih cepat diproses karena posisi citra sudah terkalibrasi.

Tahapan utama stitching meliputi:

1. Penggabungan citra horizontal dengan penyesuaian offset vertikal berdasarkan urutan pengambilan.
2. Pemotongan bagian atas citra menggunakan metode *trapezoidal masking* untuk menghilangkan distorsi tepi.
3. Rotasi korektif dengan transformasi affine untuk menyelaraskan orientasi horizontal.

Seluruh proses ini berjalan otomatis segera setelah proses pemindaian selesai dan data gambar tersedia di cloud storage.

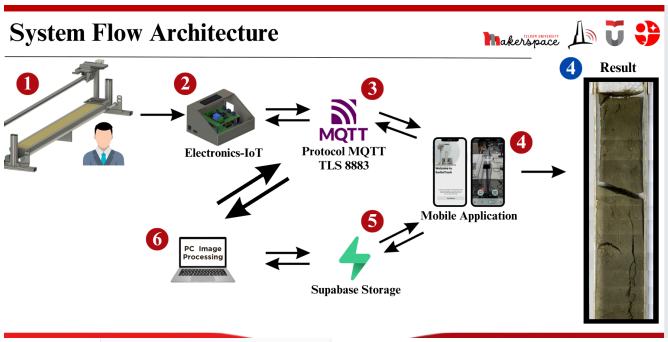
III. METODE

Bab ini menjelaskan metodologi yang digunakan dalam penelitian pengembangan sistem SedimTrack, yang dirancang untuk melakukan pemindaian sedimen secara otomatis dengan dukungan Flask, OpenCV, MQTT, dan Supabase.

Metode penelitian ini berfokus pada perancangan arsitektur sistem, implementasi proses integrasi antar komponen, serta pengujian performa sistem dalam skenario kerja nyata. Tahapan penelitian dibagi menjadi tiga bagian utama sebagai berikut

A. Tahap Perancangan System

Tahapan perencanaan sistem bertujuan untuk merancang arsitektur perangkat lunak serta alur kerja proses stitching hingga penyimpanan hasil panorama.



Gambar 1 Gambar Alur Sistem

Gambar 3.8 berfungsi sebagai ilustrasi alur sistem SedimTrack-IoT, yang menunjukkan interaksi antar komponen mulai dari proses pemindaian sedimen, pengiriman data ke cloud, pemrosesan stitching di PC, hingga pengunggahan kembali hasil panorama ke Supabase untuk ditampilkan pada aplikasi mobile.

Berdasarkan alur tersebut, perencanaan sistem perangkat lunak mencakup tiga bagian utama:

1. Perancangan Front End PC Stitching dengan Flask dan HTML

Antarmuka berbasis web dirancang menggunakan Flask sebagai backend dan HTML sebagai frontend. Antarmuka ini berfungsi sebagai pusat kontrol dan monitoring proses stitching, serta menampilkan hasil panorama yang telah diproses.

2. Perancangan Code Stitching

Proses stitching dirancang menggunakan Python dengan pustaka OpenCV. Algoritma meliputi aligning citra, cropping bagian tepi, serta rotasi korektif untuk menghasilkan panorama yang konsisten dan presisi.

3. Perancangan Supabase Storage

Supabase digunakan sebagai penyimpanan cloud terpusat. Struktur direktori disusun untuk memisahkan data mentah hasil pemindaian (raw images) dan hasil panorama (stitched images). Pengaturan akses dirancang agar PC stitching dapat mengunduh data mentah dan mengunggah kembali hasil panorama secara terstruktur dan efisien.

B. Tahap Implementasi

Tahap implementasi berfokus pada pengintegrasian seluruh komponen sistem agar dapat beroperasi secara otomatis dan sinkron, dengan memanfaatkan MQTT sebagai mekanisme komunikasi utama.

Pada tahap ini, PC stitching dihubungkan ke broker MQTT dan berlangganan (subscribe) pada topik iot/sedim/device_stat/condi. Topik ini berfungsi sebagai penanda status bahwa proses pemindaian telah selesai. Selama proses pemindaian berlangsung, PC berada dalam mode siaga sambil memantau perubahan nilai pada topik tersebut.

Ketika nilai pada topik condi berubah menjadi True, hal ini menandakan bahwa semua citra hasil pemindaian telah berhasil diunggah oleh aplikasi mobile ke Supabase Storage pada direktori /Image/Image_Upload. Setelah sinyal diterima, PC stitching secara otomatis memulai serangkaian proses berikut:

1. Mengunduh gambar mentah dari Supabase Storage.
2. Menjalankan proses stitching menggunakan Python dan OpenCV, dengan metode sequential cropping berbasis offset untuk menggabungkan citra menjadi panorama utuh.
3. Mengunggah hasil panorama ke Supabase pada direktori /Image/Results agar dapat diakses kembali oleh aplikasi mobile.

Integrasi ini memastikan seluruh alur kerja pemrosesan berlangsung otomatis tanpa intervensi manual, meminimalkan keterlambatan, dan menjaga konsistensi proses pengolahan data

IV. HASIL DAN PEMBAHASAN

Pengujian dilakukan untuk mengevaluasi kemampuan sistem dalam menggabungkan citra hasil pemindaian menjadi panorama utuh secara otomatis. Proses image stitching berjalan setelah seluruh citra berhasil diunggah ke Supabase oleh aplikasi mobile. PC stitching kemudian memproses citra menggunakan Python dan OpenCV, dengan parameter evaluasi meliputi keberhasilan stitching, waktu pemrosesan, dan kesesuaian hasil terhadap citra asli.

Uji coba dilakukan pada tiga skenario panjang lintasan: 30 cm, 50 cm, dan 100 cm. Pada lintasan 30 cm dan 50 cm, hasil stitching menunjukkan sambungan antar citra yang mulus, distribusi pencahayaan merata, dan keselarasan posisi yang konsisten. Pada lintasan 100 cm, stitching tetap berhasil secara geometris, namun muncul garis tipis pada beberapa titik sambungan akibat perbedaan pencahayaan antar segmen, bukan karena kesalahan algoritma.

Waktu pemrosesan rata-rata kurang dari 10 detik untuk seluruh skenario pada PC dengan spesifikasi standar. Pipeline otomatis dari download–stitching–upload berjalan stabil tanpa intervensi manual.

Size (cm)	Download Time (s)	Download Size (KB)	Upload Time (s)	Upload Size (KB)	Total Time (s)
30	4.40	10012	3.08	1561	3.36
50	6.55	16810	3.53	3150	4.00
100	14.46	34739	6.59	6748	7.54

Tabel 1 Performa Sistem

Hasil pengujian membuktikan bahwa sistem mampu melakukan stitching otomatis secara konsisten pada berbagai ukuran lintasan. Perbedaan pencahayaan pada lintasan panjang dapat diatasi dengan penyesuaian konfigurasi pencahayaan, sementara dari sisi algoritma, pipeline sudah stabil dan dapat diimplementasikan untuk pemantauan sedimen berbasis IoT secara real-time.

V. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem dokumentasi sedimen berbasis otomasi image stitching pada proyek SedimTrack, diperoleh kesimpulan sebagai berikut:

1. Sistem SedimTrack berhasil mengintegrasikan PC stitching berbasis Python–Flask–OpenCV dengan Supabase Storage dan MQTT**,** menghasilkan proses otomatis mulai dari pengambilan gambar, pengunggahan ke cloud, hingga stitching panorama secara penuh.
2. Protokol MQTT dengan QoS 1 dan TLS berfungsi optimal dalam sinkronisasi status antar perangkat. PC stitching dapat memulai proses secara otomatis ketika topik status condi berubah menjadi *True*, menunjukkan bahwa semua gambar telah tersedia di cloud.
3. Proses stitching pada lintasan 30 cm, 50 cm, dan 100 cm berhasil dilakukan dengan tingkat keberhasilan 100%, waktu pemrosesan di bawah 10 detik, dan hasil panorama yang konsisten secara geometris. Variasi pencahayaan pada lintasan panjang (100 cm) menjadi faktor visual minor yang dapat diperbaiki dengan penyesuaian pencahayaan.
4. Performa transfer data menunjukkan waktu unduh dan unggah yang proporsional terhadap jumlah citra, dengan rata-rata total waktu pemrosesan end-to-end tetap efisien.
5. Sistem telah memenuhi tujuan penelitian, yaitu menghasilkan pipeline dokumentasi sedimen yang otomatis, terintegrasi, dan andal, meskipun ruang pengembangan lebih lanjut masih terbuka untuk

analisis lanjutan seperti segmentasi citra atau integrasi kecerdasan buatan.

REFERENSI

[1] W. K. Chen, *Linear Networks and Systems*. Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] A. H. G. Chan dan D. S. T. Lam, "Application of IoT for environmental monitoring: A review," *Environmental Monitoring and Assessment*, vol. 192, no. 10, pp. 1–20, Oct. 2020.

[3] R. Szeliski, *Image Alignment and Stitching: A Tutorial*. San Rafael, CA: Morgan & Claypool Publishers, 2006.

[4] G. van Rossum dan F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[5] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*. Sebastopol, CA: O'Reilly Media, 2018.

[6] A. Banks dan R. Gupta, *MQTT Essentials: A Lightweight IoT Protocol*. Birmingham, UK: Packt Publishing, 2016.

[7] HiveMQ, "MQTT Essentials Part 6: Quality of Service 0, 1 & 2," Internet: https://www.hivemq.com/mqtt-essentials, Jan. 2021 [Diakses: 2 Agu. 2025].

[8] G. Bradski dan A. Kaehler, *Learning OpenCV 4: Computer Vision with Python*. Sebastopol, CA: O'Reilly Media, 2020.