STITCHING VERTICAL MENGGUNAKAN METODE SIFT PADA CITRA KULIT UNTUK DETEKSI DEFECT KULIT DI PT. USAHA LOKA

TUGAS AKHIR

CHROMATIUS ASWATAMA ISTANTO

312010004

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

MALANG

2024

LEMBAR PENGESAHAN

TUGAS AKHIR

STITCHING VERTICAL MENGGUNAKAN METODE SIFT PADA CITRA KULIT UNTUK DETEKSI DEFECT KULIT DI PT. USAHA LOKA

Oleh:

CHROMATIUS ASWATAMA ISTANTO

NIM. 312010004

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI DAN DESAIN

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom

Dosen Pembimbing 1,	Dosen Pembimbing 2,
Windra Swastika, S.Kom., MT., Ph.D	**Prof.Dr.Eng Romy Budhi Widodo**
NIP. 20070039	**NIP. 20070035**

Dekan Fakultas Teknologi dan Desain

Prof.Dr.Eng Romy Budhi Widodo

NIP. 20070035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Stitching Vertical Menggunakan Metode SIFT Pada Citra Kulit Untuk Deteksi Defect Kulit Di PT. Usaha Loka" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 14 Agustus 2024

Chromatius Aswatama Istanto
NIM. 312010004

STITCHING VERTICAL MENGGUNAKAN METODE SIFT PADA CITRA KULIT UNTUK DETEKSI DEFECT KULIT DI PT. USAHA LOKA

Chromatius Aswatama Istanto, Windra Swastika, Romy Budhi Widodo

312010004

ABSTRAK

Permasalahan utama yang dihadapi dalam industri kulit adalah proses pemeriksaan kecacatan pada kulit, yang saat ini masih bergantung pada pemeriksaan manual oleh tenaga kerja, sehingga memakan waktu dan tenaga. Berdasarkan masalah tersebut, penelitian ini dilakukan untuk mengembangkan teknologi yang dapat mengubah pemeriksaan manual menjadi pemeriksaan otomatis menggunakan citra digital. Penelitian ini bertujuan untuk mengatasi masalah dalam industri kulit, khususnya pada proses pemeriksaan kecacatan kulit yang selama ini masih dilakukan secara manual dan memakan waktu serta tenaga, dengan mengembangkan teknologi otomatis berbasis citra digital. Sistem yang dikembangkan menggunakan kamera Intel RealSense dan Blackfly S High Frame Rate untuk menangkap citra kulit kambing, sapi, dan domba, dengan penggabungan gambar secara vertikal untuk mendapatkan citra yang utuh. Hasil penelitian menunjukkan bahwa kamera Blackfly S High Frame Rate, meskipun membutuhkan waktu rata-rata 11,34 detik untuk proses penggabungan, menghasilkan citra dengan resolusi tinggi (1920 x 6037 piksel) dan tekstur yang lebih tajam, serta jumlah keypoints yang lebih banyak, seperti pada kulit domba dengan 3743 dan 3783 keypoints pada frame A dan B, dengan 638 keypoints yang cocok. Sementara itu, kamera IntelRealSense unggul dalam kecepatan pemrosesan dengan waktu rata-rata 5,57 detik dan resolusi 960 x 5031 piksel, namun menghasilkan jumlah keypoints dan kecocokan yang lebih rendah. Evaluasi dengan metode SIFT menunjukkan bahwa kamera Blackfly S lebih cocok untuk aplikasi yang memerlukan kualitas gambar tinggi, sementara IntelRealSense lebih sesuai untuk aplikasi yang membutuhkan respons cepat. Hasil ini memberikan panduan penting bagi industri dalam memilih teknologi yang tepat untuk otomatisasi deteksi cacat kulit dengan mempertimbangkan keseimbangan antara kualitas gambar dan kecepatan pemrosesan.

Kata kunci: Pengenalan Citra Kulit, industri kulit, *Image Stitching Vertical*, SIFT (*Scale-Invariant Feature Transform*), Blackfly S High Frame Rate, Intel Realsense

STITCHING VERTICAL USING THE SIFT METHOD ON LEATHER IMAGES FOR DEFECT DETECTION AT PT. USAHA LOKA

Chromatius Aswatama Istanto, Windra Swastika, Romy Budhi Widodo

312010004

ABSTRACT

The main issue faced by the leather industry is the process of inspecting defects in the leather, which currently relies on manual inspection by workers, consuming time and labor. Based on this problem, this research was conducted to develop a technology that can transform manual inspection into an automated process using digital imaging. The study aims to address the problem in the leather industry, particularly in the defect inspection process, which has so far been performed manually and is time-consuming, by developing an automated technology based on digital imaging. The system developed uses Intel RealSense and Blackfly S High Frame Rate cameras to capture images of goat, cow, and sheep leather, with vertical image stitching to obtain a complete view of the leather. The results of the study show that the Blackfly S High Frame Rate camera, although requiring an average of 11.34 seconds for the stitching process, produces high-resolution images (1920 x 6037 pixels) with sharper textures and a higher number of keypoints, such as 3743 and 3783 keypoints on frame A and B for sheep leather, with 638 matched keypoints. Meanwhile, the IntelRealSense camera excels in processing speed with an average time of 5.57 seconds and a resolution of 960 x 5031 pixels, but produces fewer keypoints and matches. Evaluation using the SIFT method shows that the Blackfly S camera is more suitable for applications requiring high image quality, while IntelRealSense is more appropriate for applications that require quick response. These findings provide crucial guidance for the industry in selecting the right technology for automated leather defect detection, considering the balance between image quality and processing speed.

Keywords: Leather Image Recognition, Leather Industry, Vertical Image Stitching, SIFT (Scale-Invariant Feature Transform), Blackfly S High Frame Rate, Intel RealSense

KATA PENGANTAR

Puji syukur dipanjatkan ke hadirat Tuhan Yang Maha Esa, yang atas segala karunia-Nya sehingga tugas akhir dengan judul "Stitching Vertical Menggunakan Metode SIFT Pada Citra Kulit Untuk Deteksi Defect Kulit Di PT. Usaha Loka" dapat terselesaikan dengan baik. Penulis ingin mengucapkan terima kasih kepada seluruh pihak yang telah membantu penulis dalam proses pembuatan tugas akhir, di antaranya:

1. Ibu Dr. Kestrilia Rega Prilianti, S.Si., M.Si., selaku ketua penguji,
2. Bapak Windra Swastika, S.Kom., MT., Ph.D, selaku dosen pembimbing I,
3. Bapak Prof. Dr. Eng. Romy Budhi Widodo, selaku dosen pembimbing II,
4. Bapak Hendry Setiawan, ST, M.Kom., selaku ketua program studi Teknik Informatika,
5. Orang tua yang telah memberikan dukungan selama proses pembuatan Tugas Akhir,
6. Ko Patrick dan Ko Kenny, selaku pengarahan dan pembimbing project penelitian ini menjadi Tugas akhir,
7. Pak Mamad dan Pak Budi, selaku pekerja pabrik yang telah membantu saya selama proses pengambilan data,
8. Serta teman-teman yang telah memberikan dukungan untuk dapat menyelesaikan Tugas Akhir,

Laporan ini disusun sebagai salah satu prasyarat kelulusan. Penulis menyadari bahwa tugas akhir ini masih memiliki beberapa kekurangan dan jauh dari sempurna. Oleh karena itu, kritik dan saran diharapkan dapat membantu memperbaiki kekurangan yang ada. Demikian tugas akhir ini dapat bermanfaat.

Malang, 14 Agustus 2024

Chromatius Aswatama Istanto

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Citra gambar adalah visualisasi objek atau fenomena dalam bentuk 2 dimensi, seperti lukisan, foto atau grafik. Dalam pengolahan citra digital, citra gambar diubah dan dianalisis menggunakan algoritma yang telah ditentukan untuk berbagai tujuan seperti perbaikan kualitas gambar, ekstraksi informasi, pengenalan objek dan kompresi objek. Banyak yang dapat dimiliki citra gambar dalam memberikan solusi untuk visualisasikan proses resolusi digital secara terukur dan akurat. Pada penelitian tentang kamera video berkecepatan tinggi digunakan dalam aplikasi ilmiah (Manin et al., 2018) menyatakan perbandingan kinerja kamera video dengan kecepatan tinggi penting dalam keberlangsungan teknologi industri dalam bidang pengolahan citra. Oleh karena itu, penggunaan kamera memiliki peran besar dalam penelitian ini karena sebuah kebutuhan dari salah satu industri kulit yang ada di Kota Malang maka penelitian ini ingin dilakukan.

Industri sepatu dan tas memanfaatkan kulit sebagai bahan utama untuk produk yang dibuat. Sebagai tantangan utama yang dihadapi adalah dalam mendeteksi cacat pada kulit secara otomatis. Saat ini, proses identifikasi cacat masih bergantung pada pemeriksaan manual oleh ahli, yang memakan waktu dan tenaga. Untuk meningkatkan efisiensi dan akurasi, perlu dilakukan deteksi otomatis menggunakan teknologi *deep learning*. Namun, untuk melatih model *deep learning* tersebut, diperlukan data *training* yang mencakup citra kulit secara utuh. Salah satu kendala dalam mengakuisisi citra kulit secara penuh adalah ukurannya yang panjang, sekitar 1.5 – 2 meter, sehingga tidak dapat diambil dengan sempurna menggunakan kamera biasa. Untuk mengatasi masalah ini, diperlukan pengembangan sistem perangkat yang mampu mendapatkan citra kulit secara utuh dengan memanfaatkan konveyor mesin. Dalam penelitian ini, akan digunakan metode penggabungan gambar secara vertikal untuk menggabungkan beberapa citra menjadi satu citra utuh. Kamera yang dipilih untuk pengambilan citra adalah Blackfly S dan Intel Sense karena kemampuannya dalam menghasilkan gambar berkualitas tinggi. Dengan mengimplementasikan solusi ini, diharapkan industri

sepatu dan tas dapat meningkatkan kualitas produknya serta efisiensi dalam proses produksi.

Oleh karena itu, pada penelitian ini dibuat sebuah perbandingan antara kamera *depth-sensing* atau kedalaman dan kamera *high-fps* yang dapat menangkap gambar berkecepatan tinggi. Pemanfaatan perbandingan ini diharapkan dapat menyelesaikan titik temu untuk mengetahui manakah alat yang efisien terhadap permasalahan yang terjadi dalam kasus tertentu atau penelitian sebelumnya. Dengan adanya penelitian ini juga diharapkan dapat membantu penelitian mengenai pengembangan aplikasi yang baru agar lebih efisien.

1.2 Identifikasi Masalah

Untuk mendapatkan citra kulit secara utuh dari kulit yang bergerak di konveyor sehingga defect kulit dapat terdeteksi, diperlukan metode penggabungan gambar baik secara vertikal maupun horizontal dari citra tangkapan kamera.

1.3 Batasan Masalah

Beberapa batasan dalam penyelesaian masalah pada penelitian ini diantaranya:

a) Pendeteksian dilakukan pada objek kulit yang diletakkan diatas konveyor berjalan tanpa tingkat kecepatan yang diatur.
b) Menggunakan 2 jenis kamera yaitu Intel RealSense depth camera dan Blackfly S High Speed camera dengan lensa 4mm.
c) Faktor lingkungan seperti kebisingan pada lokasi produksi tidak dibahas dalam penelitian ini.
d) Data yang diperoleh dari kulit tidak membahas ukuran berat massanya.
e) Dilakukan stitching secara vertikal.

1.4 Rumusan Masalah

Dalam melakukan penelitian ini, rumusan masalah yang dikemukakan adalah sebagai berikut:

1. Bagaimana menerapkan metode penggabungan gambar secara vertikal sehingga dapat disusun menjadi citra kulit yang utuh?
2. Bagaimana mencari cara kamera dengan kecepatan konveyor yang sesuai standard operasional perusahaan agar kualitas gambar tetap dapat optimal.

1.5 Tujuan

Berikut beberapa tujuan yang dibuat dalam penelitian ini sebagai berikut:

1. Menerapkan metode penggabungan gambar secara vertikal sehingga dapat disusun menjadi citra kulit yang utuh.
2. Mencari cara kamera dimana dengan kecepatan konveyor yang sesuai standard operasional perusahaan agar kualitas gambar tetap dapat optimal.

1.6 Manfaat

Penelitian ini bermanfaat sebagai referensi dalam mengidentifikasi, menyediakan rekomendasi dan meningkatkan produktivitas serta kualitas alat untuk mendapatkan citra kulit yang utuh pada mesin konveyor.

1.7 Luaran

Luaran dari penelitian ini adalah publikasi ilmiah dan aplikasi yang berbasis *desktop*.

1.8 Sistematika Penulisan

Sistematika penulisan pada laporan tugas akhir ini dibagi menjadi lima bab sebagai berikut.

1. Bab 1 Pendahuluan

   Bab pendahuluan berisikan latar belakang yang mendasari tugas akhir ini, identifikasi masalah, batasan dari masalah, rumusan masalah, tujuan yang ingin dicapai, manfaat dari penelitian, luaran dan sistematika penulisan
2. Bab 2 Tinjauan Pustaka

   Bab tinjauan pustaka berisikan teori yang mendasari pada penelitian ini, penjelasan mengenai metode yang dipakai dan alat-alat yang digunakan dalam penelitian.
3. Bab 3 Analisis dan Perancangan Sistem

   Bab analisis dan perancangan sistem berisikan rancangan sistem secara keseluruhan, alur kerja sistem, alat-alat yang digunakan dan desain skema pengambilan datat.
4. Bab 4 Hasil dan Pembahasan

   Bab hasil dan pembahasan berisikan hasil dari *stitching* vertikal secara keseluruhan, dan analisis, hasil dari pengujian dan pembahasannya, serta penjelasan mengenai kendala yang ditemui pada saat proses pembuatan.

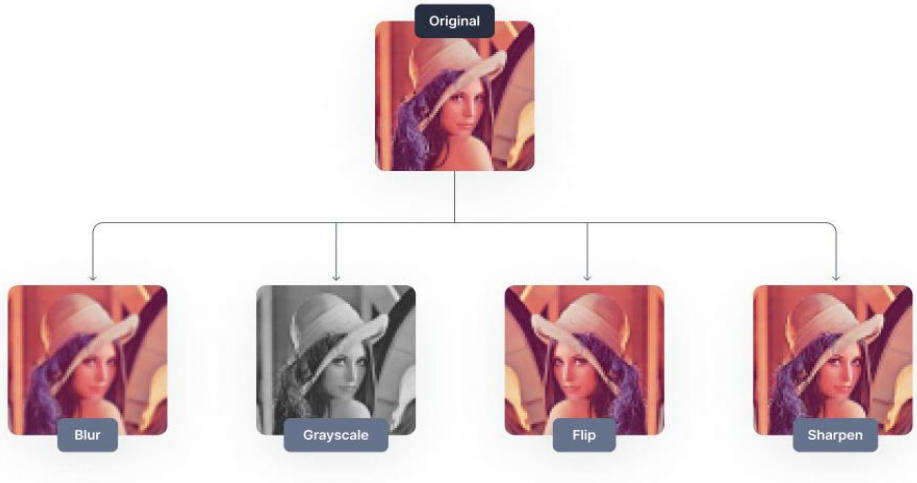
5. Bab 5 Kesimpulan dan Saran

   Bab kesimpulan berikan kesimpulan dari keseluruhan yang dikerjakan dari tugas akhir dan saran untuk penelitian berikutnya jika ingin dikembangkan.

BAB II
TINJAUAN PUSTAKA

2.1 Pengolahan Citra

Pengolahan citra merupakan bagian dari ilmu informatika yang digunakan untuk meningkatkan kualitas gambar sehingga lebih mudah diinterpretasikan oleh manusia dan komputer (Sari et al., 2017). Gambar 2.1 merupakan pengolahan citra atau *Image* yang representasi visual terdiri dari berbagai warna, pola, dan bentuk yang bisa memiliki nilai estetika yang menarik. Ini bisa berupa foto udara, potongan lintang dari objek tertentu, gambar wajah, hasil pemindaian, dan sejenisnya. Secara ilmiah, sebuah citra adalah representasi fungsi tiga dimensi (3D), yang biasanya digambarkan sebagai intensitas warna yang berfungsi dari koordinat spasial x dan y.



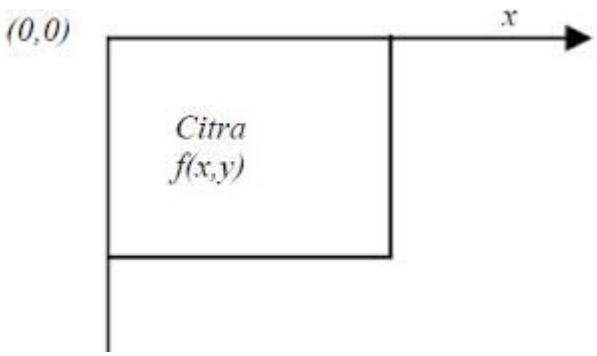
Gambar 2.1 Contoh pengolahan citra.
(sumber: https://www.v7labs.com/blog/image-processing-guide)

Pada umumnya, pengolahan citra digital terdapat tiga tahap yang perlu dilakukan ialah:

1. Memasukan gambar masukan menggunakan *image acquisition tool*
2. Menganalisis dan memanipulasi *image* yang dimasukan
3. Mendapatkan keluaran yang dapat digunakan dan merupakan hasil gambar baru dalam bentuk informasi dari gambar masukan.

2.2 Model Citra

Citra merupakan sebuah matriks dua dimensi merepresentasikan fungsi intensitas cahaya, yang mengacu pada posisi pada bidang dengan suatu fungsi intensitas cahaya yang dapat dinyatakan sebagai f(x,y), di mana f adalah nilai amplitudo pada koordinat spasial (x,y). Gambar 2.2 merupakan citra referensi menggunakan dua variabel yang menunjukkan posisi pada bidang, dan nilai amplitudo f(x,y) menggambarkan intensitas cahaya pada setiap titik koordinat (x,y) (Mulyawan et al., 2011).

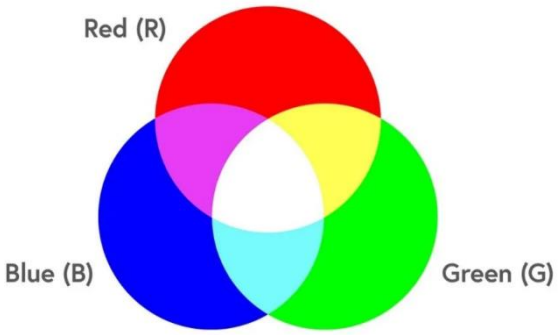


Gambar 2.2 Citra diskrit
(sumber: https://images.app.goo.gl/vK6hXSF6e2GZZz9D7)

2.3 RGB (Red-Green-Blue)

Menurut (Sari et al., 2017), Dengan menggunakan model RGB, yang berarti merah, hijau, dan biru, merupakan tiga matriks *grayscale*, matrik R untuk warna merah, matriks G untuk hijau, dan matriks B untuk biru, menunjukkan derajat kecerahan warna merah pada skala keabuan 0-255, dengan nilai 0 menunjukkan kegelapan (hitam), dan nilai 255 menunjukkan warna merah yang paling terang.

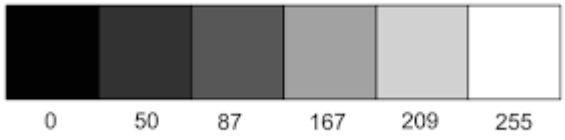
G-layer merupakan matrik yang menunjukkan tingkat kecerahan untuk warna hijau, sementara B-layer menunjukkan matrik tingkat kecerahan untuk warna biru. Dari tiga warna dicampur membentuk warna yang berbeda (Mulyawan et al., 2011).



Gambar 2.3 Komposisi warna RGB
(sumber: https://solusiprinting.com/apa-itu-model-warna-rgb-nih-penjelasan-lengkapnya/)

2.4 *Grayscale* (Derajat Keabuan)

Grayscale merupakan citra gambar digital hitam dan putih yang mempunyai setiap pikselnya berwarna gradasi mulai dari putih hingga hitam (RD. Kusmanto, 2011). Pada ambar grayscale, warna abu-abu adalah R (merah), G (hijau), dan B (biru) dengan intensitas yang sama. Oleh karena itu, intensitas gambar disimpan dalam 8 bit integer yang memberikan 256 kemungkinan, dimulai dari 0 hingga 255, dengan nilai 0 untuk hitam dan 255 untuk putih, dan nilai di antaranya adalah derajat keabuan. (Sharma, 2018). Gambar 2.4 merupakan tingkat *grayscale* dari 0 hingga 255.



Gambar 2.4 Tingkat keabuan
(sumber :https://miro.medium.com/v2/resize:fit:806/1*GdhoA4vooipUmjNm6Sh8OQ.png

2.5 Thresholding

Thresholding merupakan teknik pemisahan gambar yang mengubah gambar *grayscale* menjadi gambar berwarna hitam dan putih yang termasuk dalam *Image processing*. Proses *thresholding* ini dilakukan dengan mengubah nilai piksel yang lebih tinggi dari tingkat *threshold* ke piksel hitam 0 dan piksel yang lebih rendah

dari tingkat *threshold* ke putih 255. Tingkat *thresholding* ini dapat diatur secara manual atau otomatis menggunakan metode seperti Otsu's *method* (Jain & Petrou, 2017).

Thresholding digunakan untuk membantu dalam proses pemisahan dan penyutingan gambar, seperti memisahkan *foreground* dari *background,* memperoleh informasi yang lebih efektif dan mudah dipahami. Gambar 2.5

Gambar 2.5 Pengolahan citra *Threshold* pada kendaraan
(sumber : https://encord.com/blog/image-thresholding-image-processing/)

merupakan citra hasil dari *threshold* metode segmentasi citra yang paling sederhana dan memainkan peran penting dalam *Image processing* dengan mengekstraksi informasi penting sebuah gambar.

2.6 OpenCV

OpenCV, yang merupakan singkatan dari *Open Source Computer Vision Library,* adalah repositori gratis dan *open source* yang menyediakan kumpulan fungsi *computer vision* dan *machine learning*. OpenCV digunakan untuk mengolah gambar secara *realtime* dan efisien untuk komputasi yang intensif. *Library* ini memiliki antarmuka dalam beberapa bahasa pemrograman, yaitu C++, C, Python, Java dan MATLAB. OpenCV digunakan dalam berbagai aplikasi, termasuk pengolahan citra *factory product inspection*, *medical imaging*, analisis keamanan, interaksi manusia-mesin, pengaturan kamera, pengolahan citra stereo (3D vision), dan pengolahan citra robot (Boesch, n.d.).

2.7 *Image stitching*

Image stitching adalah proses menggabungkan beberapa gambar kecil menjadi satu gambar yang lebih besar, yang sering disebut sebagai panorama. Proses ini mencakup beberapa langkah, mulai dari deteksi titik-titik kepala (*keypoints*) dan ekstraksi deskripsi lokal yang tidak berubah (SIFT, SURF, dll.) dari dua gambar input, pencocokan deskripsi antara gambar, menggunakan algoritma RANSAC untuk menyimpulkan matriks homografi yang menggabungkan gambar, dan pemrosesan gambar menggunakan matriks homografi tersebut (Oktaviano et al., 2021).

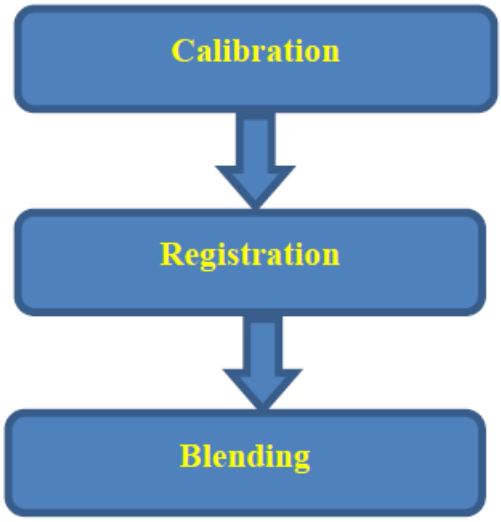
Gambar 2.6 Pengolahan citra Mosaik menjadi 6 gambar
(sumber : https://www.lumenera.com/blog/images-processing-for-microscopy-with-focus-stacking-and-image-stitching)

Image stitching digunakan dalam proses untuk menjaga fokus pada masing-masing citra yang menghasilkan gambar fokus secara keseluruhan dan maksimal. Ada 2 metode utama untuk menggabungkan data citra menjadi gambar yang lebih besar.

- Pertama, dengan menggunakan satu kamera dan memperoleh gambar secara bertahap dari waktu ke waktu.
- Kedua, dengan menggunakan beberapa kamera yang mengambil gambar secara simultan atau pada waktu yang serupa, dan kemudian menggabungkan gambar-gambar tersebut menjadi satu gambar yang lebih besar.

Gambar 2.6 merupakan proses penggabungan gambar dikenal sebagai mosaik, yang mengambil kombinasi gambar dari gambar seni artistik yang dirangkai dengan berbagai bahan. Dalam konteks pencitraan digital, mosaik merujuk pada kumpulan gambar yang digabungkan menggunakan perangkat lunak.

Image Stitching memiliki 3 tahapan komponen utama menurut Adel & Elmogy, (2014) untuk menghasilkan perkiraan parameter kamera ekstrinsik dan intrinsik dimulai dari *Calibration*, *Registration*, dan *Blending* yaitu sebagai berikut:



Gambar 2.7 Tahapan komponen *image stitching*
(sumber : https://www.researchgate.net/profile/Mohammed-Elmogy/publication/267638940_Image_Stitching_based_on_Feature_Extraction_Techniques_A_Survey/links/5456ab440cf2cf51648032c6/Image-Stitching-based-on-Feature-Extraction-Techniques-A-Survey.pdf)

2.7.1 *Calibration*

Kalibrasi gambar bertujuan untuk mengurangi perbedaan antara model lensa yang ideal dan karakteristik lensa kamera yang digunakan. Perbedaan ini muncul karena adanya cacat optik seperti distorsi dan perbedaan eksposur di antara gambar-gambar yang diambil. Proses kalibrasi melibatkan penentuan parameter intrinsik dan ekstrinsik kamera untuk merekonstruksi struktur tiga dimensi suatu pemandangan dari koordinat piksel titik gambar. Parameter ekstrinsik kamera menentukan posisi dan orientasi kamera dalam kerangka acuan dunia yang telah diketahui, sementara parameter intrinsik kamera menghubungkan koordinat piksel suatu titik gambar dengan koordinat yang sesuai dalam kerangka acuan kamera.

2.7.2 *Registration*

Registrasi gambar adalah tahapan utama dalam proses pembuatan mosaik. Tujuannya adalah untuk menciptakan kesesuaian geometris antara dua atau lebih gambar yang diambil dari sudut pandang yang berbeda. Dengan melakukan registrasi, kita dapat membandingkan gambar-gambar tersebut secara tepat dan melanjutkan proses dengan langkah-langkah selanjutnya. Registrasi gambar merupakan proses penyesuaian posisi dan orientasi dua gambar atau lebih agar dapat disatukan dengan baik. Dengan demikian, perpindahan antara gambar-gambar tersebut dapat dilakukan secara mulus dan sambungan antara keduanya dapat tercipta tanpa kecacatan atau hilang.

2.7.3 *Blending*

Blending diterapkan pada keseluruhan *stitching* untuk menciptakan transisi yang halus di antara gambar-gambar yang digabungkan. Terdapat dua metode populer untuk melakukan pencampuran gambar. Salah satunya disebut *alpha feathering blending*, dimana rata-rata tertimbang dari dua gambar diambil. Metode ini efektif ketika piksel-piksel di kedua gambar sejajar dan perbedaan utama antara keduanya adalah pergeseran intensitas keseluruhan. Pendekatan lain yang populer adalah piramida Gaussian. Dalam metode ini, gambar-gambar digabungkan pada berbagai pita frekuensi dan kemudian difilter sesuai kebutuhan. Semakin rendah pita frekuensinya, semakin kabur batasnya, sehingga piramida Gaussian menghasilkan pencampuran yang halus sambil menyamarkan batas antara gambar-gambar.

2.8 Intel RealSense D435

Gambar 2.7 merupakan Intel RealSense D435 adalah kamera kedalaman yang menggunakan teknologi *stereoscopic* untuk menangkap gambar kedalaman. Kamera ini memiliki sensor RGB *global shutter* dan sensor kedalaman *stereoscopic*, yang memungkinkan pengguna untuk menangkap gambar yang akurat dan dapat digunakan dalam aplikasi seperti robotika, *Augmented Reality* (AR), dan *Virtual Reality* (VR). Kamera ini memiliki kemampuan sensor yang lebar (87° × 58°) dan memiliki kemampuan output kedalaman sampai dengan 1280 × 720 piksel. Kamera ini juga memiliki kemampuan *output* RGB sampai dengan 1920 × 1080 piksel dan *frame-rate* maksimal 30 fps. Kamera ini dapat digunakan dalam lingkungan

indoor/outdoor dan memiliki kemampuan *range* sampai dengan 10 meter. Kamera ini juga didukung oleh SDK Intel RealSense yang dapat digunakan di Windows, Linux, dan platform lainnya (*Intel® RealSense™ Depth Camera D435*, n.d.).

Gambar 2.8 Intel RealSense D435
(sumber: https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html)

Tabel 2.1 Spesifikasi Intel RealSense D435
(sumber: https://www.intelrealsense.com/depth-camera-d435/)

Features	***Use environment:*** Indoor/Outdoor	***Ideal range:*** 0.3m to 3m
Depth	***Depth technology:*** Stereoscopic ***Minimum depth distance (Min-Z) at max resolution:*** ~28 cm ***Depth Accuracy:*** <2% at 2 m[1]	***Image sensor technology:*** Global Shutter ***Depth Field of View (FOV):*** 87° × 58° ***Depth output resolution:*** Up to 1280 × 720 ***Depth frame rate:*** Up to 90 fps

RGB	***Frame resolution:*** 1920 × 1080 ***Frame rate:*** 30 fps	***Sensor FOV (H × V):*** 69° × 42° ***Sensor resolution:*** 2 MP ***Sensor technology:*** Rolling Shutter
Major Components	***Camera module:*** Intel RealSense Module D430+ RGB Camera	***Vision processor board:*** Intel RealSense Vision Processor D4
Physical	***Form factor:*** Camera Peripheral ***Length × Depth × Height:*** 90 mm × 25 mm × 25 mm	***Connectors:*** USB-C* 3.1 Gen 1* ***Mounting mechanism:*** – One 1/4-20 UNC thread mounting point. – Two M3 thread mounting points.

2.9 Blackfly S High Frame rate color

Blackfly S merupakan kamera yang dikeluarkan oleh Teledyne FLIR yang dilengkapi dengan sensor canggih seperti Complementary Metal-Oxide-Semiconductor (CMOS), pengaturan waktu ekspos yang *real-time*, dan pengaturan segmentasi warna *threshold*. Kamera ini dirancang untuk aplikasi *machine vision* dan dapat membantu *monitoring* operasional industri. Ukurannya yang kecil ideal untuk digunakan dalam proses/aktivitas industri dengan hasil dokumentasi yang tepat, beresolusi, dan berkualitas tinggi. Termasuk juga kontrol manual maupun otomatis atas hasil dokumentasi dan *pre-processing* pada kamera.

Gambar 2.9 BFS-U3-23S3C-C USB 3.1 Blackfly® S, Color Camera (sumber: https://www.edmundoptics.com/p/bfs-u3-23s3c-c-usb3-blackflyreg-s-color-camera/41347/

Tabel 2.2 Spesifikasi Blackfly® S, Color Camera BFS-U3-23S3C-C (sumber: https://www.flir.asia/products/blackfly-s-usb3/?model=BFS-U3-23S3C-C&vertical=machine+vision&segment=iis)

Features	***Acquisition Modes:*** *Continuous* / *Single Frame* / *Multi Frame*	***Exposure Range:*** 5 µs to 30 *seconds*
	Flash Memory: *6* MB *non-volatile memory*	***Exposure Control:*** *Hardware trigger, programmable via camera API, automatic*
	Gain Control: *Manual, automatic* (AGC)	***Image Buffer:*** 240 MB
	Image Processing: *Color correction matrix, gamma, lookup table, saturation, and sharpness*	***Max Frame Rate Standard:*** 163
	Partial Image Modes: Pixel binning, decimation, ROI	***User Sets:***

		2 *user configuration sets for custom camera settings*
	Synchronization : *Software trigger, hardware trigger, action command*	***White Balance:*** Manual, automatic (AWB)
Performance	***Absolute Sensitivity Threshold (y):*** 4.81	***Resolution:*** 1920 x 1200
	Gain Range: 0 dB to 47dB	***Dynamic Range dB:*** 71.45
	***Quantum Efficiency Blue*:** 49.48 (460nm)	***Saturation Capacity (e-):*** 10772
	Quantum Efficiency Green: 60 (530nm)	***Temporal Dark Noise (Read Noise):*** 2.38
	Quantum Efficiency Red: 48.03 (625nm)	
Physical	***Auxiliary Output:*** 3.3 V, 120 mA maximum	***Connector:*** *USB3 Micro-B | Hirose HR10A-7R-6PB*
	Data Interface: Gigabit Ethernet 1 Gbps	***Digitization Bit Depth (ADC):*** 10-bit, 12-bit
	Dimensions [W x H x L]: 29 mm × 29 mm × 30 mm	***Mass:*** 36 g
	Lens Mount: *C-mount*	***Megapixels:*** 2.3
	Non-Isolated I/O Ports: *1 bi-directional, 1 input*	***Operating Temperature:*** 0° to 50°C
	Operating Humidity: 20 to 80% (*no condensation*)	***Pixel size:*** 3.45

Part number: BFS-U3-23S3C-C	***Power Requirements:*** 8-24 V via GPIO or 5 V via USB3
Power Consumption <3 W	***Spectrum:*** *Color*
Sensor Format: 1/2.3"	***Storage Humidity:*** 30 to 95% *(no condensation*)
Sensor Model: Sony IMX392	***Storage Temparature:*** -30° to 60°C
Sensor Type: CMOS	***Optical Filter:*** *IR-Cut Off (670 nm)*
Opto-isolated I/O Ports: *1 input, 1 output*	

2.10 Blackfly Lenses

Blackfly Lenses adalah jenis lensa *fisheye* yang dikembangkan oleh TTArtisan, sebuah produsen lensa. Lensa ini memiliki sudut pandang yang lebih luas dari lensa *fisheye* biasa, yang memungkinkan pengambilan gambar dengan sudut pandang yang lebih unik dan kualitas gambar yang tinggi (Tabatabaei et al., 2020). Ada 4 lensa yang dipakai dalam penelitian ini yaitu sebagai berikut.

Gambar 2.10 Fixed Focal Length Lens 4mm UC Series
(sumber: https://www.edmundoptics.com/p/4mm-uc-series-fixed-focal-length-lens/2966/)

Tabel 2.3 Spesifikasi 4mm UC Fixed Focal Length lens
(sumber: https://www.edmundoptics.com/p/4mm-uc-series-fixed-focal-length-lens/2966/)

Komponen	Keterangan
***Max Horizontal* (FOV)**	23.1mm - 84.8°
***Max Vertical* (FOV)**	15.7mm - 65.3°
Working Distance (mm)	0 - Infinity
Aperture	f1.8 - f11
Focal Length	6mm, Fixed
Vendor	Edmund Optics
Product model	Fixed Focal Length Lens
Imaging Lens Type	*High Performance Lens with Compact Form Factor*
Storage Temperature (°C)	*-20 to +60 For questions regarding operating temperature please contact our support team*

Gambar 2.11 Fixed Hik Robot FA Lens 6mm
(sumber :https://www.hikrobotics.com/en/machinevision/productdetail?id=4899&pageNumber=1&pageSize=50

Tabel 2.4 Spesifikasi 6mm lens FA MVL-HF0624M-10MP
(sumber :https://www.hikrobotics.com/en/machinevision/productdetail?id=4899&pageNumber=1&pageSize=50)

Komponen	Keterangan
***Max Horizontal* (FOV)**	7.38mm - 62.46°
***Max Vertical* (FOV)**	4.92mm - 44.05°
Working Distance (mm)	0.1m (min)
Aperture	f2.8 - f16
Focal Length	6mm, Fixed
Vendor	HIK Robot
Product model	MVL-HF0624M-10MP
Imaging Lens Type	-
Storage Temperature (°C)	-

Gambar 2.12 Fixed Focal Length Lens 12mm C Series
(sumber : https://www.edmundoptics.com/p/12mm-c-series-fixed-focal-length-lens/14949/)

Tabel 2.5 Spesifikasi 12mm C Series Fixed Focal Length Lens
(sumber : https://www.edmundoptics.com/p/12mm-c-series-fixed-focal-length-lens/14949/)

Komponen	Keterangan
***Max Horizontal* (FOV)**	41.4°
***Max Vertical* (FOV)**	31.4°
Working Distance (mm)	100 - ∞
Aperture	F1.8 - f16

Komponen	Keterangan
Focal Length	12mm, Fixed
Vendor	Edmund Optics
Product model	Fixed Focal Length Lens
Imaging Lens Type	*Compact Lens*
Storage Temperature (°C)	*-20 to +60 For questions regarding operating temperature please contact our support team*

Gambar 2.13 Fixed Focal Length Lens 12mm f/4 Cr Series
(sumber : https://www.edmundoptics.com/p/12mm-f4-cr-series-fixed-focal-length-lens/32976/)

Tabel 2.6 Spesifikasi 12mm f/14C Series Fixed Focal Length Lens
(sumber : https://www.edmundoptics.com/p/12mm-f4-cr-series-fixed-focal-length-lens/32976/)

Komponen	Keterangan
***Max Horizontal* (FOV)**	81.2mm - 41.2°
***Max Vertical* (FOV)**	60.5mm - 31.3°
Working Distance (mm)	98 - ∞
Aperture	f/4
Focal Length	12mm, Fixed
Vendor	Edmund Optics
Product model	Fixed Focal Length Lens
Imaging Lens Type	-
Storage Temperature (°C)	*-20 to +60 For questions regarding operating temperature please contact our support team*

2.11 Visual Studio Code

Visual Studio Code (VS Code) adalah sebuah editor kode sumber yang dikembangkan oleh Microsoft dan dapat dijalankan pada berbagai sistem operasi, termasuk Windows, macOS, dan Linux. Editor ini dirancang untuk memudahkan pengembangan aplikasi dengan fitur-fitur yang lengkap dan ekosistem yang luas. Selain mendukung berbagai bahasa pemrograman seperti Python, JavaScript, C++, dan banyak lagi, VS Code juga dilengkapi dengan ekstensi yang memungkinkan pengembang untuk menyesuaikan lingkungan kerja sesuai kebutuhan mereka. Fitur-fitur seperti *debugging,* kontrol versi Git, integrasi terminal, dan IntelliSense (penyelesaian kode otomatis) membuat VS Code menjadi alat yang sangat efisien dan produktif.

Gambar 2.14 Logo Visual Studio Code
(sumber : https://code.visualstudio.com/brand)

2.12 Algoritma SIFT (*Scale-Invariant Feature Transform*)

Algoritma SIFT (*Scale-Invariant Feature Transform*) adalah sebuah metode untuk mengidentifikasi dan menggambarkan fitur-fitur lokal dalam citra digital (Saputra, 2023). Diciptakan oleh David Lowe pada tahun 2004, algoritma ini telah menjadi populer dalam berbagai aplikasi seperti pengenalan objek, wajah, dan sidik jari. SIFT bekerja dengan mencari titik-titik yang memiliki karakteristik unik dan stabil, seperti bentuk, ukuran, dan orientasi, dalam citra. Titik-titik ini, dikenal sebagai *keypoints*, digunakan untuk memberikan deskripsi pada citra dan membandingkan citra yang serupa.

Menurut Saputra (2023), algoritma SIFT terdiri dari beberapa bagian, salah satunya adalah sebagai berikut:

2.12.1 *Gaussian* dan *Difference of Gaussian Scale Space*

Untuk membuat skala ruang skala Gaussian dalam algoritma SIFT, fungsi Gaussian digunakan. Persamaan yang menunjukkan konvolusi gambar dengan kernel Gaussian pada berbagai tingkat skala dilihat pada rumus (1) .

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma} e^{-(x^2-y^2)/2\sigma^2} \quad (2.1)$$

Keterangan:

- e = 2,72 (nilai ketetapan)
- σ = parameter (skala) dalam jumlah kabur
- π = 3,14

Setelah space skala Gaussian diperoleh, kemudian dicari L(x, y, σ), yang merupakan konvolusi gambar asli I (x, y) dengan filter Gaussian pada skala σ, seperti yang ditunjukkan dalam Persamaan (2).

$$L(x, y, \sigma) = G(x, y, \sigma) \times I\ (x, y) \quad (2.2)$$

Untuk menemukan perbedaan Gaussian (DoG) pada skala dengan menggunakan persamaan (3).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \times I\ (x, y) \quad (2.3)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Dimana:

- σ adalah parameter (skala) jumlah kabur. Semakin besar nilainya, semakin blur.
- x , y adalah koordinat titik citra.
- $L(x, y, k\sigma)$ adalah konvolusi citra asli $I\ (x, y)$ dengan Gaussian filter $G(x, y, \sigma)$ pada skala k σ dengan nilai $k = \sqrt{2}$.

2.12.2 Deteksi Ekstremum

Proses deteksi ekstremum (nilai maksimum dan minimum) dalam algoritma SIFT dilakukan dengan membandingkan nilai setiap piksel pada skala ruang DoG (*Difference of Gaussians*) dengan tetangganya (Saputra, 2023). Titik-titik penting dipilih jika nilainya lebih besar atau lebih kecil daripada semua tetangganya yang berjumlah 26 piksel. Selanjutnya, *keypoint* yang tidak stabil disaring. *Keypoint* yang tidak stabil dapat berasal dari suara atau kontras yang rendah, terutama jika

posisinya berada di tepi gambar. Untuk menghilangkan *keypoint* yang tidak stabil, batas kontras ditetapkan, yang berarti jika nilai kontras di bawah batas tersebut, titik tersebut tidak akan menjadi *keypoint*.

2.12.3 Penentuan Orientasi *Keypoint*

Setiap gambar sampel L(x,y) memiliki magnitudo m(x,y) dan orientasi θ(x,y) dihitung dengan perbedaan piksel seperti yang ditunjukkan dalam Persamaan, dan nilai skala *keypoint* ditentukan dengan menggunakan gambar Gaussian rata-rata L yang memiliki nilai yang mendekati skala *keypoint* (4).

$$m\,(x,y) = \sqrt{\begin{pmatrix} L\,(x+1,y) - L\,(x-1,y)^2\; + \\ (L(\,x,y+1)\; -\; L(x,y-1))^2 \end{pmatrix}} \qquad (2.4)$$

$$\theta\,x,y\; = \arctan\left(\frac{L(x,y+1) - L(x,y+1)}{L(x+1,y) - L(x-1,y)}\right)$$

Keterangan:

- $m(x,y)$ = magnitude terhadap koordinat lokasi x dan y
- $\theta(x,y)$ = orientasi terhadap koordinat lokasi x dan y

2.12.4 Penentuan Deskriptor Lokal

Menghitung vektor *deskriptor* untuk setiap *keypoint* adalah langkah terakhir dalam algoritma SIFT (Saputra, 2023). Proses ini dilakukan pada gambar yang memiliki skala paling dekat dengan skala *keypoint* tersebut. Pertama, orientasi diperoleh dengan menggunakan piksel 4x4 dengan presisi 8 bit di sekitar setiap tombol. Kemudian, untuk menghitung magnitudo dan nilai orientasi pada sampel dalam wilayah 16x16 di sekitar tombol, dibuat *histrogram* orientasi. Dengan parameter "*half-width*", yang setara dengan setengah lebar deskriptor, magnitude dihitung dengan fungsi Gaussian. Deskriptor kemudian menjadi vektor dari semua nilai histogram. Vektor ini memiliki 128 elemen dengan 16 histogram, masing-masing dengan presisi 8 bit.

2.13 *Feature Based Registrations*

Feature Based Registrations merupakan proses registrasi dalam pengolahan citra yang menggunakan fitur-fitur khusus (seperti sudut, tepi, atau area lain yang unik) dari citra untuk melakukan pencocokan dan registrasi antar citra (BODA, 2009). Teknik ini bertujuan untuk mengoreksi pergeseran, rotasi, dan skala antar citra agar mereka dapat di-*align* dengan benar. *Feature Based Registration* biasanya

melibatkan beberapa tahap, termasuk deteksi fitur, pencocokan fitur, dan estimasi transformasi. Fitur-fitur ini dapat ditemukan menggunakan algoritma seperti SIFT *(Scale-Invariant Feature Transform)* atau SURF *(Speeded-Up Robust Features)*, dan kemudian dipasangkan antara dua citra yang akan diregistrasi. Estimasi transformasi, seperti homografi atau transformasi afinitas, kemudian digunakan untuk memetakan satu citra ke citra lainnya. Teknik ini sering digunakan dalam aplikasi seperti *image stitching*, *augmented reality*, dan registrasi citra medis.

2.14 *Feature Detection*

Feature detection adalah proses dalam pengolahan citra digital yang melibatkan deteksi dan ekstraksi fitur-fitur lokal pada gambar (Rosebrock, 2018). Fitur-fitur ini bisa berupa berbagai karakteristik seperti bentuk, ukuran, orientasi, dan warna, yang digunakan untuk mengidentifikasi objek atau mengenali citra. Dalam konteks pengenalan wajah, *feature detection* digunakan untuk menemukan dan mengenali fitur-fitur seperti mata, hidung, dan mulut untuk mengidentifikasi individu. Sedangkan dalam pengenalan objek, teknik ini digunakan untuk mendeteksi fitur-fitur objek seperti bentuk dan ukuran untuk mengenali objek secara keseluruhan (Kurnia et al., 2022). Sementara dalam pengenalan citra, *feature detection* membantu dalam menemukan dan mengidentifikasi fitur-fitur citra seperti bentuk dan warna untuk memahami isinya (Fan & Liu, 2015). Dengan aplikasi yang luas dalam pengenalan gambar, video, atau bahkan citra 3D, *feature detection* memiliki peran penting dalam berbagai aplikasi pengolahan citra digital.

2.15 Feature Matching

Untuk menentukan area penyambungan yang tepat, fitur yang diperoleh dari metode SIFT dicocokkan dengan fitur ini dilakukan dengan menggunakan metode jarak geometri n-dimentional (Beis & Lowe, n.d.).

$$d\,(p,q) = \sqrt{(p_1 - q_1)^2 + (p_1 - q_1)^2 + \cdots + (p_n - q_n)^2} \qquad (2.5)$$

Dengan menggunakan rumus Jarak Euclidean, jarak yang menunjukkan seberapa besar perbedaan antara fitur-fitur dapat dihitung. Selanjutnya, dengan menetapkan ambang batas, dapat ditentukan apakah suatu fitur dianggap cocok. Dalam beberapa kasus, beberapa fitur terletak di titik yang sama atau sangat dekat, sehingga jaraknya relatif kecil dengan fitur dari gambar lain. Oleh karena itu, kedua fitur tersebut perlu dipertimbangkan sebagai kandidat fitur yang cocok. Untuk

mengatasi masalah ini, metode K-Nearest Neighbour dapat digunakan, di mana pencarian dilakukan hingga ditemukan sejumlah k titik dengan jarak terbaik.

Umumnya, pencocokan fitur dilakukan dengan membandingkan semua fitur dalam satu gambar dengan semua fitur dalam gambar lain (*Brute Force Search*), yang memakan waktu cukup lama. Untuk meningkatkan efisiensi, digunakan *k-Dimensional Tree* dengan metode *Best Bin First Search*.

2.16 *Nearest Neighbor Search* (NNS)

Nearest Neighbor Search (NNS) adalah suatu metode yang digunakan untuk menemukan titik terdekat dari suatu titik *query* dalam suatu set data. Metode ini digunakan dalam berbagai aplikasi, seperti pengembangan aplikasi absensi *face recognition*, pengembangan aplikasi absensi *face recognition* menggunakan OpenCV library dan algoritma SIFT, dan pengembangan aplikasi absensi face recognition menggunakan OpenCV library dan algoritma SIFT (Lifshits, 2010).

Metode *Nearest Neighbor Search* melibatkan beberapa langkah, seperti:

1. *Preprocessing*: untuk mengubah format data menjadi lebih sesuai dengan kebutuhannya.
2. *Indexing*: digunakan membuat indeks untuk memudahkan pencarian data.
3. *Query*: untuk mencari data yang sesuai dengan kebutuhan.
4. *Retrieval*: Mengembalikan data yang ditemukan.

2.17 Sistem Stabilitas Gambar Optik (OIS)

Sistem Stabilitas Gambar Optik (OIS) atau *Body Image Stabilization* (IBIS) adalah teknologi yang digunakan pada kamera untuk mengurangi efek guncangan dan gerakan pada gambar yang diambil. OIS dan IBIS berfungsi dengan cara menghitung pergerakan lensa dan mengontrolnya secara otomatis ketika mengambil gambar atau video, sehingga dapat mengurangi foto yang buram dan menghasilkan resolusi gambar yang tajam (Neal et al., 2024).

Dalam beberapa aplikasi, OIS dan IBIS digunakan untuk mengatasi perbedaan dalam sudut pandang, jarak, dan waktu antara citra-citra yang diregistrasi. Contohnya, dalam aplikasi pengembangan sistem informasi wisata, metode ini digunakan untuk menggabungkan citra-citra yang diambil dari sudut pandang yang berbeda untuk membantu wisatawan menemukan lokasi yang lebih efektif.

2.18 Penelitian Terdahulu

Penelitian sebelumnya dilakukan pada (Romdhoni et al., 2020) tentang meningkatkan dan menentukan kebutuhan kacang kedelai dengan kualitas tinggi. Dengan menggunakan Decision Tree, penelitian ini mengekstraksi dan mengklasifikasikan ciri dari citra digital kedelai kuning menggunakan teknik GLCM *(Gray-Level Co-Occurrence Matrix*), hasil pengujian menunjukkan tingkat akurasi mencapai 95% untuk data dari kamera 1 (Oppo F7) dan 90% untuk kamera 2 (Samsung A20) dengan jumlah data latih 96 citra (80%) setelah dua puluh empat kali uji coba. Temuan ini menjanjikan penggunaan teknologi dalam penilaian kualitas kedelai secara efisien dan akurat.

Penelitian lain tentang penggabungan gambar (*image stitching*) dibawah air menggunakan metode SIFT (Scale-invariant feature transform) (Zhang et al., 2023). Penelitian ini dirancang untuk meningkatkan kinerja citra gambar dibawah air yang memiliki titik fitur yang buram/kabur dan tanpa bergantung pada detektor lainnya. Dalam metode ini menggunakan teknik fusi *fade-in* dan *fade-out* untuk menciptakan gambar panorama bawah air yang koheren. Hasil eksperimen menunjukkan bahwa metode yang diusulkan menunjukkan peningkatan ketahanan, terutama dalam skenario di mana mendeteksi titik fitur menjadi sulit, jika dibandingkan dengan metode SIFT tradisional. Selain itu, metode ini mencapai akurasi pencocokan yang lebih tinggi dan menghasilkan hasil yang berkualitas lebih tinggi dalam penyatuan gambar bawah air.

Penelitian lain yang pernah dilakukan oleh (Mohammadi et al., 2024) meneliti tentang penggunaan teknik *image stitching* dengan menggunakan mikroskopis untuk menggabungkan beberapa gambar mikroskopis yang tumpang tindih dari sampel biologis. Prosesnya melibatkan dua langkah utama yaitu: registrasi berpasangan dan penyelarasan. Penelitian ini melakukan analisis perbandingan terperinci terhadap berbagai teknik registrasi berpasangan dalam hal waktu eksekusi dan kualitas hasilnya. Hasilnya menunjukkan bahwa metode berbasis fitur, terutama fitur SURF, lebih efektif dalam mengatasi tantangan pencahayaan yang tidak merata pada gambar mikroskopis.

Adapun penelitian membahas tentang menyelidiki potensi penggunaan paralelisasi *Message Passing Interface* (MPI) untuk meningkatkan kecepatan pada

proses penggabungan citra (Ramadhan et al., 2024). Dengan membagi tugas-tugas penggabungan di antara beberapa node prosesor, penelitian ini bertujuan untuk mempercepat proses tanpa kekurangan kualitas sedikit pun. Hasilnya menunjukkan bahwa penggunaan MPI dapat mengurangi waktu penyambungan gambar secara signifikan, dari waktu 1506,63 detik tanpa MPI menjadi 346,8 detik dengan memakai empat node pada komputer. Dengan adanya potensi baru yang baik dan efisien waktu maka dapat menjadi bahan mengeksplorasi teknik pengoptimalan tambahan dan mengevaluasi dampaknya terhadap kecepatan dan kualitas dalam aplikasinya.

Penelitian lain tentang teknologi *Image stitching* yang digunakan saat malam hari yang bertujuan untuk menggabungkan serangkaian gambar dengan sudut pandang berbeda dalam suasana malam yang terjadi, dengan tujuan menciptakan citra panorama bidang panjang meluas dan menyeluruh (Yan et al., 2023). Hasil eksperimen menunjukkan ada sebuah peningkatan kualitas gambar dalam hal informasi, kontras, dan penekanan *noise* dibandingkan dengan algoritma lainnya. Selain itu, metode ini berhasil mengekstrak fitur garis yang penting dari gambar malam hari, memudahkan proses penggabungan gambarnya.

Penelitian yang dilakukan oleh (Liu et al., 2023) pernah meneliti tentang penggunaan metode *multi-scene Image Stitching* yang dikenal sebagai MSIS yang disebut *Merge-sorting image stitching*. Metode ini diusulkan untuk mengatasi masalah kompleksitas dan waktu komputasi yang lebih tinggi dalam penggabungan gambar. MSIS menggabungkan gambar secara langsung berdasarkan urutan gabungan gambar tanpa memerlukan pencocokan semua pasangan gambar. Hasil eksperimen menunjukkan bahwa MSIS menghasilkan pengurangan waktu komputasi sekitar 10% dibandingkan dengan model berbasis Binary-tree, serta mengurangi distorsi pada gambar yang digabungkan.

Penelitian yang dilakukan oleh (Sutjipto et al., 2010) meneliti tentang penggabungan citra panorama secara otomatis menggunakan metode *invariant feature*. Penelitian tersebut memuat proses penggabungan beberapa gambar citra foto yang tumpang tindih dengan bidang untuk menghasilkan gambar panorama dan gambar beresolusi tinggi. Salah satu teknik yang digunakan adalah SIFT (*Scale Invariant Feature Transform*), dapat mengidentifikasi dan menjelaskan fitur lokal

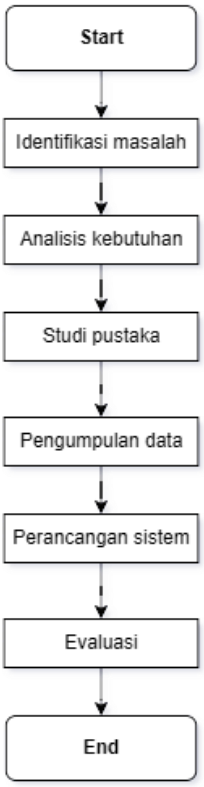
gambar dengan ketahanan terhadap penskalaan, rotasi, perubahan sudut pandang, intensitas pencahayaan, dan *noise* pada citra. Hasil dari penelitian tersebut menunjukkan metode yang efektif untuk mengatasi perbedaan skala, rotasi, dan perbedaan cahaya yang masuk.

Penelitian ini dapat dioptimalkan dengan mengadopsi beberapa temuan dari penelitian terdahulu. Romdhoni et al. (2020) menunjukkan bahwa penggunaan GLCM untuk ekstraksi fitur dan Decision Tree untuk klasifikasi dapat meningkatkan akurasi penilaian kualitas, sehingga metode ini dapat diterapkan dalam penelitian untuk meningkatkan akurasi deteksi defect pada kulit. Zhang et al. (2023) menggunakan teknik fusi *fade-in* dan *fade-out* dengan metode SIFT untuk meningkatkan ketahanan dan kualitas penggabungan gambar, yang juga dapat diadopsi untuk meningkatkan hasil *stitching* citra kulit. Mohammadi et al. (2024) menemukan bahwa metode berbasis fitur seperti SURF efektif dalam menangani pencahayaan tidak merata, yang relevan untuk aplikasi pada citra kulit dengan kondisi pencahayaan yang bervariasi. Ramadhan et al. (2024) menunjukkan bahwa paralelisasi dengan MPI dapat mempercepat proses stitching tanpa mengorbankan kualitas citra, sehingga penggunaan MPI dapat meningkatkan efisiensi dalam penelitian ini. Yan et al. (2023) mengembangkan teknik yang meningkatkan kontras dan mengurangi *noise* dalam penggabungan gambar dimalam hari, yang dapat diterapkan untuk meningkatkan kualitas gambar dalam kondisi pencahayaan rendah. Liu et al. (2023) mengusulkan metode MSIS yang mengurangi waktu komputasi dan distorsi, sehingga pendekatan ini dapat digunakan untuk efisiensi waktu dalam proses *stitching*. Sutjipto et al. (2010) menunjukkan bahwa metode SIFT efektif untuk deteksi dan deskripsi fitur lokal yang tahan terhadap variasi skala, rotasi, dan pencahayaan, sehingga metode ini sangat relevan untuk diterapkan dalam penelitian deteksi defect pada kulit. Dengan mengintegrasikan teknik-teknik ini, penelitian dapat meningkatkan akurasi, efisiensi, dan kualitas dalam mendeteksi defect pada kulit.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Alur Penelitian

Penelitian dimulai dengan mengidentifikasi masalah yang terjadi pada pabrik PT. Usaha Loka, Malang. Dari masalah yang dibahas dalam pabrik untuk menghasilkan penerapan *image stitching* secara vertikal secara utuh pada mesin konveyor yang berjalan. Gambar 3.1 merupakan keseluruhan alur penelitian yang dilakukan. Alur penelitian yang dilakukan disajikan dalam Gambar 3.1.



Gambar 3.1 Alur penelitian

3.2 Identifikasi Masalah

Pada penelitian ini, perusahaan pabrik pengolahan bahan tekstil kulit hewan menghadapi kendala dalam memperoleh proses pemeriksaan citra kulit yang cacat agar lebih mudah menentukan apakah kulit tersebut cacat atau tidak. Informasi yang didapat menunjukkan bahwa hampir keseluruhan bahan kulit yang diperoleh mengalami cacat dan sulit untuk diperiksa, karena selama ini pemeriksaan hanya dilakukan secara manual dengan mata kepala sendiri saat proses *finishing* di pabrik. Untuk mengatasi permasalahan ini, penelitian ini difokuskan pada pengembangan

dan implementasi metode yang efektif untuk menggabungkan gambar-gambar. Proses ini akan menggunakan dua kamera canggih dan tinggi, yaitu Intel Realsense Depth dan Blackfly S High Frame Rate, untuk memastikan akurasi dan kecepatan dalam penggabungan serta analisis citra.

3.3 Analisis Kebutuhan

Pada perancangan dan pembuatan sistem penggabungan data citra gambar berbasis video dari kulit yang berjalan pada mesin konveyor dengan bantuan algoritma SIFT, diperlukan analisis kebutuhan yang mendalam. Analisis kebutuhan ini mencakup kebutuhan perangkat keras dan perangkat lunak baik dari sisi pabrik maupun peneliti. Tujuan dari analisis ini adalah untuk memastikan bahwa penelitian dan implementasi sistem dapat berjalan dengan baik dan lancar.

3.3.1 Kebutuhan Pabrik

Kebutuhan analisis ini diperoleh berdasarkan identifikasi masalah yang telah dijelaskan pada sub-bab sebelumnya, yaitu proses pemeriksaan kulit menggunakan dua kamera canggih, yaitu Intel Realsense Depth dan Blackfly S High Frame Rate. Proses ini memanfaatkan metode *image stitching* untuk menggabungkan beberapa citra yang dihasilkan dari kedua kamera tersebut sehingga menjadi satu citra vertikal yang utuh. Pengembangan metode image stitching ini bertujuan untuk mempermudah proses pemeriksaan kulit, baik saat proses pengerjaan di pabrik maupun oleh pengguna yang memanfaatkan produk akhirnya.

Berikut ini adalah beberapa perangkat keras yang sudah disediakan untuk digunakan oleh peneliti dalam menguji dan memanfaatkan alat yang telah disediakan untuk penelitian ini.

1. Perangkat Keras
    a. Kamera Intel Realsense Depth D435
    b. Kamera Blackfly S High Frame Rate BFS-U3-23S3C-C
    c. Blackfly Lenses @4 bh (4mm UC, 12mm f/4Cr, 12mm C, 6mm FA Lens)
    d. Kabel SS USB 3.0 Super High Speed AWM Style
    e. Kabel USB 3.1 Type-C Super High Speed AWM Style
    f. Mesin Konveyor

g. Lampu LED Neon 120cm
h. Kabel rol stop kontak 4 lubang

2. Resources dokumen pendukung
   a. Link Github Intel Realsense Setup & Stitching.zip – TANI-main
   - https://drive.google.com/file/d/1mt-2eyYqeQZFGAtsCZR2fLspAccp2436/view?usp=drive_link
   b. Link rincian hardware: https://docs.google.com/spreadsheets/d/1N02GPUgGZzOxVcbN0Quu2qm9EAmEXZ5BfTiuwYnDR04/edit?usp=sharing

3.3.2 Kebutuhan Peneliti

Berikut ini adalah beberapa perangkat keras dan perangkat lunak yang digunakan oleh peneliti dalam melakukan penelitian ini.

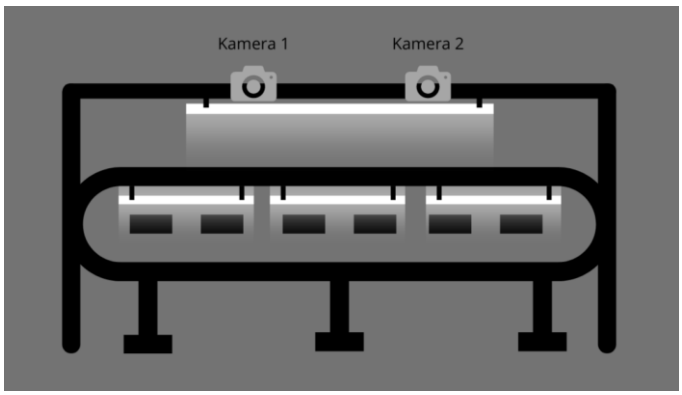
1. Perangkat keras
   a. Laptop
      - Intel Core i5-9300H 4-Core 2.4GHz
      - RAM: 8 GB
      - OS: Windows 11 Home 64 bit
      - GPU: NVIDIA GeForce GTX 1650
      - SSD: 500 GB
      - Adaptor Charger Asus TUF FX505GT 150W
      - Mouse Logitect M331
   b. Lux Meter Smart Sensor AS803 (Pengukur cahaya)
   c. Kabel Rol terminal 6 Lubang @15m
   d. Obeng EZREN 1 set EZ-0363 115in
2. Perangkat Lunak
   a. Visual Studio Code (Python versi 3.11.9)
   b. OBS Studio
   c. Intel Realsense Viewer
   d. Spinnaker SDK

3.4 Studi pustaka

Tahap ini bertujuan untuk mencari dan mempelajari referensi-referensi penelitian terdahulu serta memahami segala aspek permasalahan yang ada. Tujuannya adalah untuk mengeksplorasi metode dan alat yang digunakan, serta membandingkan metode dan teori-teori dasar dalam mengatasi masalah yang dihadapi. Teori pendukung yang dipelajari pada tahap ini *meliputi Scale-Invariant Feature Transform* (SIFT), *Image Stitching*, *Blending, Feature Matching, Feature Descriptor Extraction, Transformation Estimation*, serta perangkat keras seperti Intel RealSense D435 Depth dan Blackfly S High Frame Rate. Selain itu, penggunaan perangkat lunak seperti OpenCV juga menjadi fokus dalam tahap ini.

3.5 Pengumpulan data

Untuk pengumpulan data akan menggunakan dua kamera: Intel Realsense Depth dan Blackfly S High Frame Rate. Proses pengambilan data dilakukan kurang lebih sebanyak 5-6 kali pengambilan video untuk setiap jenis citra kulit. Setiap pengambilan video mencakup rentang *frame* antara 300-900 *frame* pervideonya. Data citra kulit yang diambil mengacu pada Gambar 3.2 Skema Mesin dan Gambar 3.3 Prototipe Posisi Kedua Kamera. Sebelum mengambilan video, dilakukan pengaturan kamera dan pencahayaan pada lingkungan. Kamera dieksekusi secara manual data foto disimpan dalam bentuk format .raw dan .bmp, sedangkan data video disimpan dalam format .avi dan .mov.

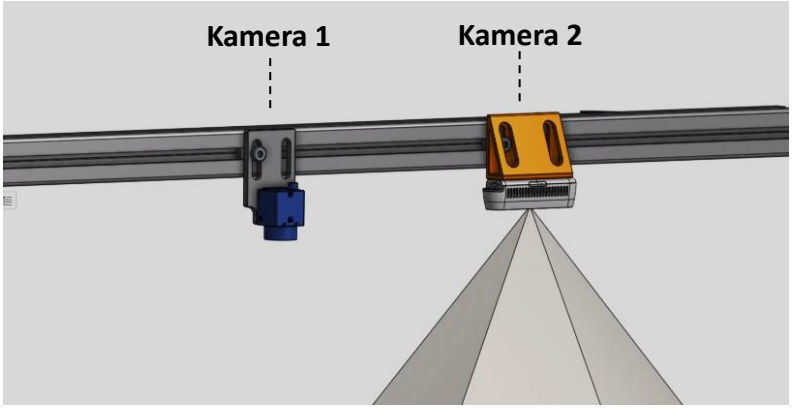


Gambar 3.2 Skema mesin dan alat operasional

Keterangan:

1. **Kamera 1**: Blackfly S High Frame Rate BFS-U3-23S3C-C
2. **Kamera 2**: InteRealsense Depth D435

Pada Gambar 3.2 skema mesin dan alat operasional menunjukkan bagaimana objek dipindahkan melalui konveyor yang dilengkapi dengan dua kamera inspeksi yang ditempatkan di atasnya.



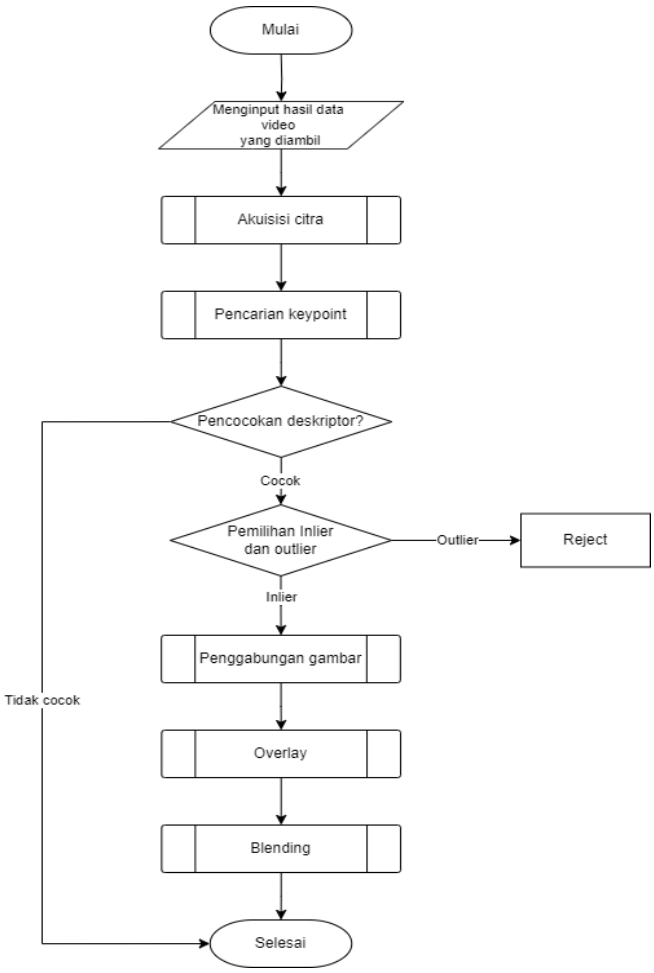
Gambar 3.3 Prototipe posisi kedua kamera

Untuk Gambar 3.3 menggambarkan secara lebih rinci prototipe dan posisi kedua kamera tersebut, memastikan bahwa setiap bagian objek terinspeksi secara menyeluruh dan data yang dihasilkan dapat dianalisis dengan baik.

3.6 Perancangan sistem

Pada proses pembuatan sistem yang sesuai dibutuhkan sebuah alur sistem model yang dibuat untuk memenuhi aplikasi yang akan dijalankan seperti Gambar 3.4 Secara berurutan, proses dimulai dengan menginput hasil data video yang diambil, di mana data tersebut dimasukkan ke dalam sistem untuk diproses lebih lanjut. Sistem kemudian melakukan akuisisi citra dari data video yang telah diinput, yang melibatkan pemisahan *frame* gambar yang akan diproses. Selanjutnya, sistem mencari *keypoint* atau titik-titik kunci pada gambar, yang merupakan fitur-fitur yang dapat diidentifikasi dan digunakan untuk mencocokkan gambar satu dengan yang lain. Deskriptor yang mewakili *keypoint* kemudian dicocokkan antara gambar satu dengan gambar lainnya untuk menentukan apakah *deskriptor* tersebut cocok atau tidak. Jika *deskriptor* cocok, proses dilanjutkan ke langkah pemilihan *inlier* dan *outlier*, di mana sistem memisahkan *inlier* (titik-titik kunci yang *valid*) dari outlier (titik-titik kunci yang tidak valid). Titik-titik kunci yang dianggap *outlier* akan di-*reject* atau ditolak, sementara yang dianggap *inlier* akan diproses lebih lanjut. Setelah inlier dipilih, gambar-gambar yang valid digabungkan dengan teknik *image stitching* atau penggabungan gambar yang akan digabungkan beberapa

frame yang dipilih setelah itu menggunakan teknik overlay untuk menyelaraskan tumpang tindih frame. Kemudian, teknik blending diterapkan untuk memastikan gambar-gambar tersebut menyatu dengan baik dan halus. Akhirnya, gambar-gambar yang telah melalui proses blending digabungkan menjadi satu gambar utuh yang merepresentasikan data video yang diambil. Proses perancangan sistem selesai pada tahap ini.



Gambar 3.4 Desain sistem

3.6.1 Akuisisi citra

Tahapan proses akuisisi citra dimulai dengan pengambilan data rekaman video menggunakan dua kamera, yaitu Blackfly S High Fps dan Intel Realsense depth. Setelah pengambilan rekaman video selesai, hasil rekaman video tersebut diproses dan diubah menjadi format JPG. Proses ini bertujuan untuk menghasilkan

citra yang dapat dengan mudah diproses dan digunakan untuk berbagai keperluan, seperti analisis atau penyimpanan lebih lanjut.

3.6.2 Pencarian *keypoint*

Pencarian *keypoint* merupakan tahap krusial dalam pengolahan gambar, di mana fitur-fitur khusus dalam gambar diidentifikasi dan digunakan sebagai titik referensi untuk menghubungkan dan menggabungkan gambar. Fitur ini dapat berupa sudut, tepi, atau bagian unik lainnya dalam gambar yang mudah dikenali. *Keypoint* ini sangat penting untuk memastikan bahwa gambar-gambar yang akan digabungkan memiliki titik-titik yang sama atau serupa, memungkinkan proses *stitching*

3.6.3 Penggabungan gambar

Penggabungan gambar melibatkan penggabungan fisik dari beberapa gambar yang telah diurutkan. Proses ini mencakup penempelan, penyesuaian posisi, dan penataan gambar berdasarkan *frame-frame* yang telah dipilih, serta menggunakan titik-titik fitur yang telah ditemukan pada tahap pencarian keypoint. Tujuannya adalah untuk membentuk satu gambar panorama utuh secara vertikal. Tahapan ini dilakukan untuk menyatukan gambar secara kasar, sehingga bisa dilanjutkan ke tahapan penyempurnaan lebih lanjut seperti overlay dan blending.

3.6.4 *Overlay*

Tahap overlay adalah langkah tambahan setelah penggabungan gambar untuk memastikan bahwa transisi antara gambar-gambar yang digabungkan mulus dan konsisten. Pada tahap ini, overlay diukur dan disesuaikan untuk menerima hasil stitching. Proses ini melibatkan penyesuaian nilai kecerahan, warna, dan detail pada area yang tumpang tindih untuk menciptakan tampilan yang lebih halus dan integrasi yang lebih baik antara gambar-gambar yang digabungkan. Penyesuaian ini penting untuk menghilangkan ketidakcocokan visual yang mungkin muncul akibat perbedaan pencahayaan atau perspektif antara gambar-gambar yang digabungkan.

3.6.5 *Blending*

Tahapan *Blending* adalah proses akhir dalam penggabungan gambar, di mana beberapa gambar disatukan menjadi satu kesatuan yang halus dan terintegrasi. Pada tahap ini, area tumpang tindih antara gambar diidentifikasi dan nilai piksel digabungkan menggunakan teknik seperti *linear* atau *pyramid blending*. Hal ini

dilakukan untuk memastikan transisi yang mulus antara gambar, menghilangkan batas-batas yang terlihat, sehingga menghasilkan gambar panorama berkualitas tinggi yang tampak seperti satu gambar utuh. Blending sangat penting untuk memastikan hasil akhir yang seragam dengan detail dan elemen visual yang terintegrasi dengan sempurna.

3.7 Evaluasi

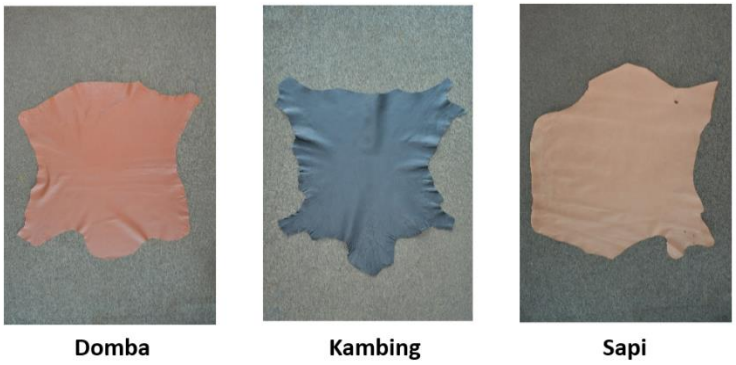
Berdasarkan hasil dari metode yang digunakan, evaluasi terhadap alat dengan dua kamera perlu dilakukan dengan mempertimbangkan parameter yang akan digunakan untuk menilai kualitas gambar dari kedua kamera tersebut meliputi resolusi dan detail ketajaman gambar. Penilaian resolusi akan difokuskan pada jumlah piksel yang dimiliki oleh kamera, di mana jumlah piksel yang lebih tinggi biasanya menunjukkan kemampuan kamera untuk menangkap lebih banyak detail. Selain itu, ketajaman gambar juga menjadi fokus utama evaluasi. Ketajaman gambar merujuk pada sejauh mana detail dalam gambar terlihat jelas dan tidak kabur. Untuk memastikan kualitas gambar yang optimal dan tajam, resolusi minimal yang dianjurkan adalah 1920 x 1080 piksel. Resolusi ini dianggap cukup tinggi untuk menghasilkan gambar dengan detail yang cukup tajam, sehingga gambar dapat digunakan secara efektif untuk berbagai aplikasi tanpa kehilangan kualitas visual.

Dalam evaluasi ini, kedua kamera akan dibandingkan secara langsung untuk melihat bagaimana perbedaan jumlah piksel dan ketajaman gambar yang dihasilkan oleh masing-masing kamera. Selain itu, hasil stitching dari citra yang diambil oleh masing-masing kamera juga akan dievaluasi. Namun, perlu dicatat bahwa hasil *stitching* memiliki keterbatasan karena tidak mengambil seluruh bagian citra. Hanya tepi objek kulit yang digabungkan, sehingga terdapat kemungkinan beberapa detail penting di tengah objek tidak tercover dengan baik. Keterbatasan ini disebabkan oleh alat yang hanya dapat mengambil gambar dari satu posisi, yaitu dari sisi tepi objek kulit saja.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Deskripsi data

Data yang dikumpulkan dalam penelitian ini terdiri dari tiga jenis citra kulit, yaitu kulit kambing, sapi, dan domba. Contoh gambar kulit kambing, sapi dan domba mengacu pada Gambar 4.1. Kulit yang digunakan kurang lebih ukuran panjang 1,5 meter dengan lebar 2 meter. Pengambilan data dilakukan dengan menggunakan dua jenis kamera, yaitu kamera Intel Realsense D435 dan Blackfly S High Frame rate tipe BFS-U3-23S3C-C, yang menghasilkan foto dan video.



Gambar 4.1 Tiga jenis kulit yang diambil

Untuk kamera Intel Realsense, terdapat dua jenis folder berbeda karena perbedaan *software* yang dipilih. Pengambilan data menggunakan perangkat lunak OBS menghasilkan video dengan format .mov, di mana setiap jenis citra kulit diambil sebanyak 6-10 kali. Sementara itu, penggunaan perangkat lunak Intel Realsense Viewer SDK menghasilkan video dengan format .bag, di mana setiap jenis citra kulit diambil sebanyak 2-3 kali, namun hanya untuk pengetesan alat.

Kemudian kamera Blackfly, data dibagi menjadi folder foto dan video untuk setiap jenis citra kulit. Pada folder foto, pengambilan dilakukan sebanyak dua kali per jenis citra kulit dengan format .bmp dan .raw. Pada folder video, pengambilan dilakukan sebanyak 6-10 kali dengan format .avi.

Total semua data yang diambil adalah 50,6 GB. Kamera Blackfly tipe BFS-U3-23S3C-C menghasilkan resolusi 1920x1200 dengan kecepatan 116,44 *frame* per detik, dibantu dengan lensa *fixed local* berukuran 4mm tipe UC series dari Edmund Optic. Kamera Intel Realsense tipe D435 depth menghasilkan resolusi

960x540 dengan kecepatan 60 *frame* per detik. Gambar 4.2 merupakan Kamera dipasang pada ketinggian sekitar 15 cm di atas mesin konveyor.

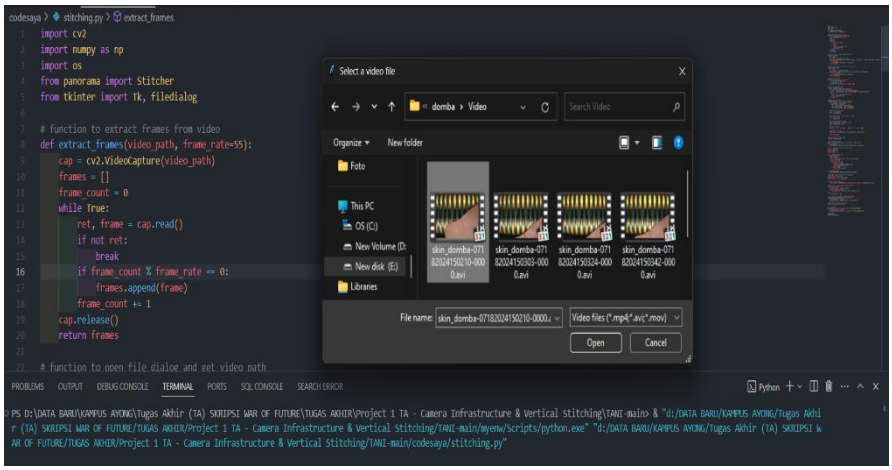
Gambar 4.2 Posisi kedua kamera ketinggian 15cm

Pengambilan video dengan kamera Blackfly berdurasi 3-6 detik dengan ukuran file sekitar 1-2 GB per video. Setiap pengambilan foto per jenis citra kulit memiliki ukuran file sekitar 8,80 MB, dengan total data sebesar 6,50 GB hingga 10 GB untuk setiap jenis kulit. Pengambilan video dengan kamera Intel Realsense menggunakan perangkat lunak OBS berdurasi 3-4 detik dengan ukuran file sekitar 42-80 MB. Pengambilan video dengan perangkat lunak Intel Realsense Viewer SDK berdurasi 4-5 detik dengan ukuran file sekitar 400 MB hingga 1 GB lebih, namun karena perangkat lunak ini kurang praktis, data dari *software* OBS akan lebih sering digunakan.

4.2 Proses pengolahan data

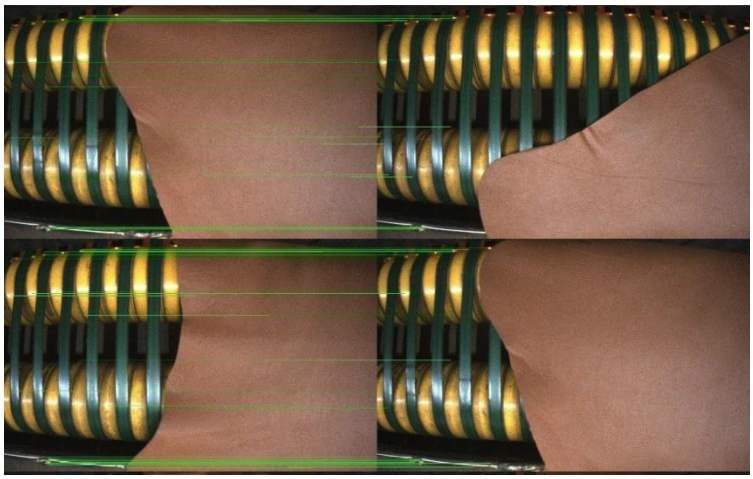
Pengolahan data proses dimulai dari beberapa tahap yang terstruktur. Pertama, pengguna mengatur nilai FPS (*frame per second*) untuk proses ekstraksi *frame* dari video, serta menentukan nilai *overlay* dan *blending* yang sesuai dengan FPS yang diinginkan. Nilai *overlay* diatur antara 100-350, dan nilai blending diatur antara 1.0-2.0 berdasarkan kecepatan FPS yang dipilih. Setelah pengaturan ini selesai, video yang akan diproses dipilih menggunakan dialog file yang memungkinkan pengguna untuk memilih file video yang diinginkan. Setelah video dipilih, video tersebut diputar dan *frame-frame* individu diambil setiap 55 *frame*

persecond dari 116.44 *frame persecond* kecepatan FPS video aslinya untuk mengurangi jumlah data yang harus diproses, sehingga mempercepat waktu pemrosesan dan mengurangi beban komputasi. Gambar 4.3 merupakan proses pemilihan video untuk nanti di ambil *frame* tertentu. Proses pengolahan video dimulai berdasarkan keutuhan kulit dan detail yang memberikan visualisasi atau informasi tentang bagaimana *frame by frame* tersebut digabungkan berdasarkan titik-titik fiturnya.



Gambar 4.3 Proses Input data video

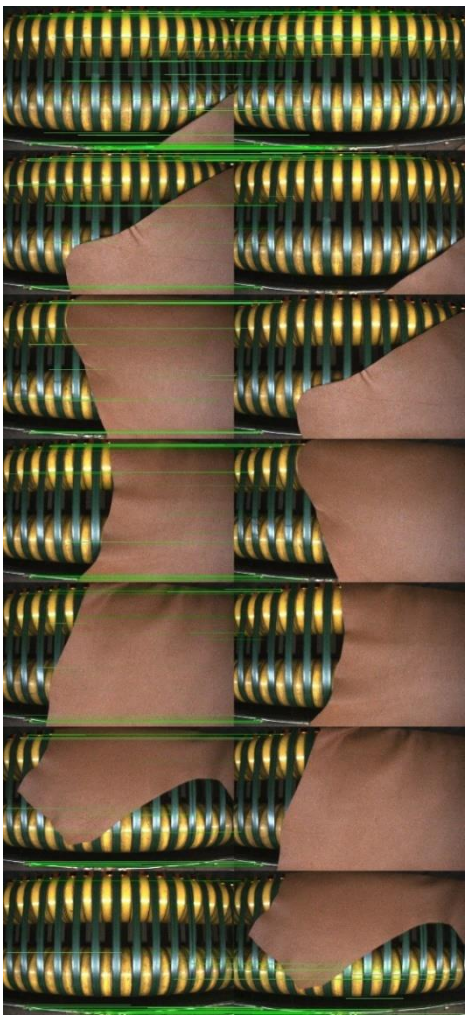
Setiap *frame* yang diambil kemudian digabungkan dua per dua secara berurutan untuk membentuk gambar panorama vertikal. Proses ini melibatkan identifikasi titik-titik fitur di antara dua *frame* dan menggabungkannya berdasarkan kesesuaian fitur-fitur tersebut. Setelah dua frame digabungkan, hasilnya akan dibersihkan dari area hitam yang mungkin muncul di tepi gambar untuk



Gambar 4.4 Proses Identifikasi titik fitur

memastikan hasil akhir yang utuh. Gambar 4.4 merupakan identifikasi titik fitur antara dua frame dan lebih.

Setelah semua *frame* digabungkan menjadi beberapa gambar yang lebih besar, gambar-gambar ini disatukan menjadi satu panorama vertikal panjang. Untuk memastikan transisi yang halus antara gambar-gambar ini, area yang tumpang tindih (*overlay*) digabungkan menggunakan metode *blending*, yang menciptakan efek peralihan yang lembut dan mulus. Selain itu, untuk memvisualisasikan proses penggabungan *frame-frame* tersebut, dibuat gambar yang menunjukkan titik-titik fitur yang cocok (*keypoint matches*) antara frame-frame yang digabungkan. Gambar 4.5 merupakan gambar *keypoint matches* kemudian digabungkan secara vertikal, memberikan pandangan yang komprehensif dan informatif tentang keseluruhan proses *stitching*.



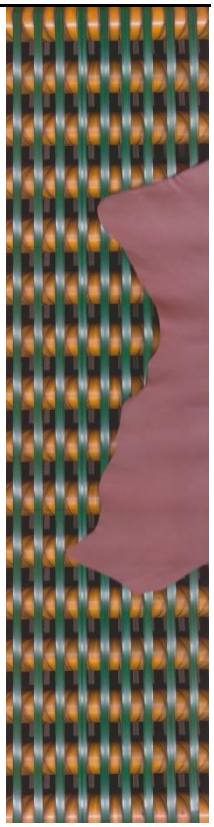
Gambar 4.5 Proses pencocokan *keypoint matches*

Setelah semua gambar panorama dan pencocokan *keypoint matches* selesai diproses, hasilnya ditampilkan pada layar. Kemudian akan muncul opsi pemilihan direktori untuk menyimpan hasil kedua gambar terdiri "*vertical_keypoint"* dan *"vertical_panorama"* dengan format .jpg, disimpan dalam direktori yang dipilih.

4.3 Hasil *Stitching* vertikal

Perbedaan resolusi kamera diatur tinggi untuk keperluan perbandingan. Tabel 4.1 merupakan hasil evaluasi hasil *stitching* antara kamera Blackfly dan Intel RealSense pada kulit di atas konveyor. Hasil *stitching* ini menunjukkan perbedaan dalam kualitas gambar yang dihasilkan, detail yang diperoleh, serta waktu yang dibutuhkan untuk menyelesaikan proses *stitching*.

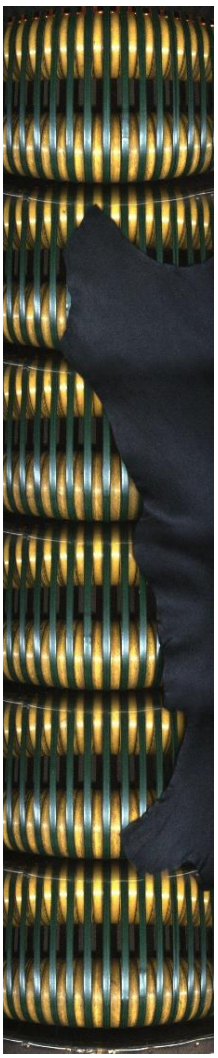
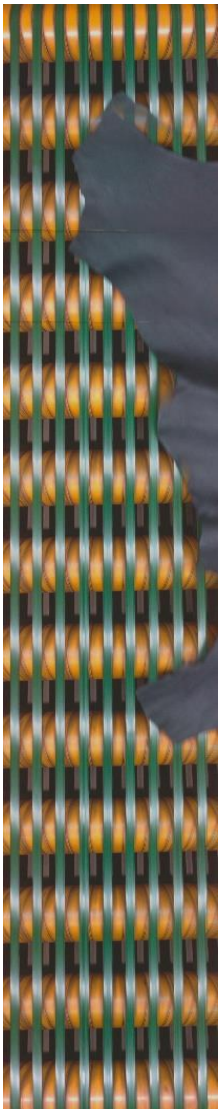
Tabel 4.1 Hasil evaluasi stitching vertikal

Kulit	**Durasi**		**Kulit Utuh**	**Hasil Stitching**	
Citra	**BlackFly**	**Intel**		**BlackFly**	**Intel**
Domba	**12'98**	**5'99**	 BlackFly	 55 fps	 15 fps

Kulit	Durasi		Kulit Utuh	Hasil Stitching	
Citra	BlackFly	Intel		BlackFly	Intel

IntelRealsense OBS

IntelRealsense
Viewer SDK

Kulit	Durasi		Kulit Utuh	Hasil Stitching	
Citra	BlackFly	Intel		BlackFly	Intel
Kambing	**10’37**	**5’06**	 Blackfly  IntelRealsense OBS  IntelRealsense Viewer SDK	 55 fps	 15 fps

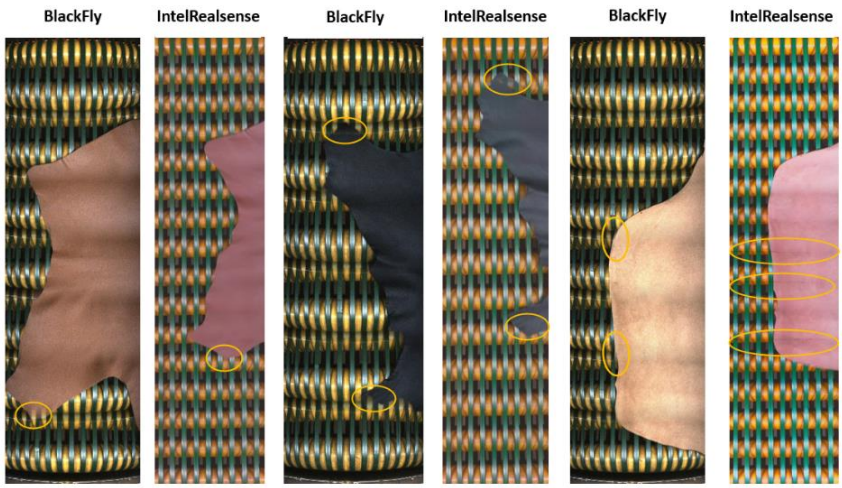
Kulit	Durasi		Kulit Utuh	Hasil Stitching	
Citra	BlackFly	Intel		BlackFly	Intel
Sapi	**10’67**	**5’50**	 Blackfly  IntelRealsense OBS  IntelRealsense Viewer SDK	 55 fps	 15 fps

Lamanya durasi selama proses hingga mendapat hasil *output stitching* dipengaruhi oleh *frame per second* (fps) dan kualitas video yang dilihat dari besaran resolusi. Saat pengambilan video IntelRealsense menggunakan 60 fps dan saat proses *stitching* menggunakan 15fps. Semakin kecil fps yang diproses maka semakin banyak *frame* yang di-*capture* saat proses *stitching* dilakukan untuk menjaga kualitas gambarnya. Namun, jika pengambilan video dilakukan dengan fps tinggi seperti pada kamera Blackfly yaitu 116 fps dan saat proses *stitching* menggunakan 55 fps, maka jumlah *frame* yang di-*capture* lebih sedikit karena kualitas gambar yang sudah optimal sejak awal pengambilan video. Oleh karena itu, perhitungan fps yang tepat penting untuk memastikan hasil *stitching* yang optimal dan menjaga kualitas video.

Besaran resolusi dalam pengambilan video menggunakan kedua kamera tersebut juga mempengaruhi lamanya durasi *stitching*. Resolusi yang lebih tinggi seperti yang dihasilkan oleh kamera Blackfly yaitu 1920x1200px memberikan detail yang lebih baik dan hasil *stitching* yang lebih berkualitas dibandingkan dengan IntelRealsense dengan resolusi yang lebih rendah yaitu 960x540px.

4.4 Analisis Hasil

Hasil yang didapatkan menunjukkan bahwa proses SIFT membutuhkan pencocokan titik *keypoint* dan pemrosesan pendekatan titik piksel pada citra *frame* yang diseleksi. Sebelum melakukan proses *stitching,* tentunya citra yang diambil perlu mengatur konfigurasi yang berbeda dalam hal resolusi, *frame rate*, dan pengaturan yang dapat membuat hasil yang signifikan.



Gambar 4.6 Hasil dari *stitching*

Berdasarkan hasil *stitching*, Gambar 4.6, pada kulit domba, BlackFly menunjukkan detail serat dan ketebalan kulit yang lebih baik dengan jumlah titik *feature keypoint* yang lebih banyak, sedangkan IntelRealsense menghasilkan gambar yang lebih halus dengan warna yang sedikit berbeda. Pada kulit kambing, BlackFly menampilkan warna yang lebih kontras dan detail serat yang lebih tajam, sementara IntelRealsense menghasilkan gambar dengan warna yang lebih pudar dan tekstur yang lebih halus. Untuk kulit sapi, BlackFly memberikan representasi warna yang lebih akurat dan menampilkan tekstur permukaan yang lebih jelas. Di sisi lain, IntelRealsense menghasilkan gambar dengan tekstur yang lebih datar dan warna yang lebih cerah. Secara keseluruhan, BlackFly unggul dalam hal jumlah titik *feature keypoint* dan kualitas detail gambar, sementara IntelRealsense lebih efisien dalam hal kecepatan pemrosesan meskipun dengan jumlah titik *feature keypoint* yang lebih sedikit.

Berdasarkan hasil *stitching* yang didapatkan, titik *feature keypoint* dan *matched* atau kecocokan yang didapatkan dari keseluruhan *frame* adalah sebagai berikut:

Tabel 4.2 Hasil perbandingan titik *keypoint* dan *matched*

Kamera	**Domba**			**Kambing**			**Sapi**		
	Keypoint FrameA	Keypoint FrameB	Matched	Keypoint FrameA	Keypoint FrameB	Matched	Keypoint FrameA	Keypoint FrameB	Matched
Blackfly (55 fps) 7-8@f	3743	3783	638	4337	4371	894	4859	4811	809
Intel Realsense (15 fps) 13-16@f	1636	1691	692	2147	2141	731	3478	3452	877
Durasi	Blackfly	12'98		Blackfly	10'37		Blackfly	10'67	
	Intelreal	5'99		Intelreal	5'06		Intelreal	5'50	

Dari hasil analisis perbandingan titik *keypoint* dan *matched* mengacu pada tabel 4.2, terlihat bahwa kamera Blackfly cenderung menghasilkan lebih banyak *keypoint*, dengan total 25,904 pada kedua *frame* dibandingkan dengan IntelRealsense yang menghasilkan total 14,545 *keypoint*. Namun, jumlah *keypoint* yang cocok (*matched*) lebih tinggi pada IntelRealsense untuk objek Domba (692)

dan Sapi (877), sedangkan untuk objek Kambing, Blackfly menghasilkan lebih banyak *keypoint* yang cocok (894). Durasi pemrosesan menggunakan IntelRealsense secara konsisten lebih cepat dibandingkan dengan Blackfly untuk semua objek. Hal ini menunjukkan bahwa meskipun Blackfly menghasilkan lebih banyak *keypoint*, IntelRealsense lebih efisien dalam memproses dan mencocokkan *keypoint* dalam waktu yang lebih singkat.

Cara mengoperasikan kamera untuk pengambilan video dari beberapa kamera yang ada memerlukan penggunaan software yang berbeda-beda sesuai dengan jenis kameranya, sebagai berikut:

- Mengoperasikan Kamera Blackfly dengan *Software* SpinView

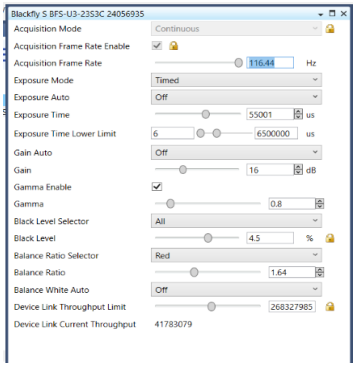
1. Unduh dan instal SpinView dari *website* resmi (https://www.flir.com/products/spinnaker-sdk/?vertical=machine+vision&segment=iis)
2. Hubungkan Kamera: Hubungkan kamera Blackfly ke komputer menggunakan kabel USB atau GigE. Pastikan kamera terdeteksi oleh SpinView. Kamera yang terhubung akan muncul di panel perangkat.
3. Pilih kamera yang ingin digunakan dari daftar “Devices” yang terhubung, kemudian Klik tulisan “Blackfly (nomor seri)” yang terhubung, kemudian Klik “Play” atau ikon “Play” yang ada window disebelahnya.
4. Atur parameter kamera seperti resolusi, *frame rate, exposure, gain*, dan *white bal*ance melalui tab "Camera Settings" atau "Device Control" mengacu pada gambar 4.7 . Pastikan *frame rate* diatur tinggi untuk menjaga kualitas video, dan pastikan pergerakan kulit diatas konveyor dari arah bawah ke atas.
5. Klik “General” kemudian ada kolom “Save Filename” untuk memilih direktori lokasi penyimpanan. Kemudian di bawahnya ada tab "Videos", disitu pilih *Video Recording Type* menjadi “Uncompressed” dan *Frame rate* Klik “Use Camera Frame Rate”.
6. Kemudian klik tombol "Start Recording" untuk memulai perekaman video. Sebelum direkam pastikan beberapa pengaturan telah di atur dengan benar.

7. Setelah itu, klik tombol "Stop Recording" saat kulit diatas konveyor sudah melewati penuh. Kemudian menunggu hingga keterangan “Of Images left in buffer” menjadi 0 karena masih proses menyimpan.

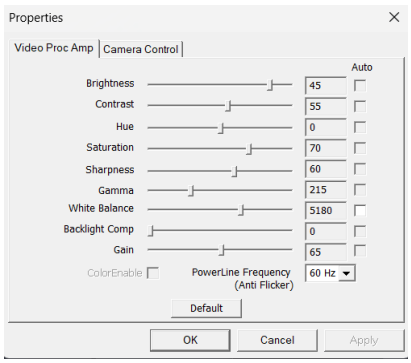
➢ Mengoperasikan Kamera IntelRealsense dengan OBS Studio

1. Unduh dan instal OBS Studio dari *website* resmi (https://obsproject.com/)
2. Hubungkan kamera IntelRealsense ke komputer menggunakan kabel USB. Pastikan kamera terdeteksi oleh sistem operasi komputer Anda.
3. OBS Studio, pada panel "*Sources*", klik tombol "+" dan pilih "Video Capture Device".
4. Pada jendela "Properties", pilih kamera IntelRealsense dari daftar perangkat yang tersedia. Mengatur pengaturan resolusi dan *frame rate* sesuai kebutuhan mengacu pada gambar 4.8, dan pastikan pergerakan kulit di atas konveyor dari arah bawah ke atas, lalu klik "OK" ,
5. Klik "Settings" di kanan bawah OBS Studio. Pada tab "Output", pilih folder yang ingin menyimpan rekaman video dan atur format file video menjadi .mov.
6. Klik tombol "Start Recording" di panel kontrol utama OBS Studio untuk memulai perekaman video.
7. Setelah selesai merekam, klik tombol "Stop Recording" saat kulit diatas konveyor sudah melewati penuh. Video akan secara otomatis disimpan ke lokasi yang telah Anda tentukan sebelumnya.

Kamera Blackfly dengan resolusi 1920x1200 piksel dan 116,44 fps menghasilkan gambar dengan detail yang sangat baik dan transisi yang mulus antara gambar yang digabungkan.



Gambar 4.7 Pengaturan Blackfly SpinView SDK



Gambar 4.8 Pengaturan IntelRealsense OBS

Ini disebabkan oleh kemampuan kamera ini untuk menangkap detail yang halus dan penyesuaian cahaya yang optimal, seperti yang terlihat pada Gambar 4.7 pengaturan ukuran *gamma* 0.8, *gain* 16 dan *white balance* yang disesuaikan secara manual. Kemudian Gambar 4.8 pengaturan kamera Intel Realsense ukuran *gamma* 215, *gain* 65 dan *white balance* nya disesuaikan menjadi 5180 karena memiliki pencahayaan yang kurang masuk.

Sebaliknya, pada pengaturan kamera Intel RealSense, yang diatur pada resolusi 960x540 piksel dan *frame rate* 60,00 fps, menunjukkan beberapa distorsi pada hasil *stitching*. Meskipun prosesnya lebih cepat, detail gambar yang dihasilkan kurang baik dibandingkan dengan Blackfly.

Hal ini terlihat pada area tumpang tindih yang masih menunjukkan perbedaan yang jelas, yang mungkin disebabkan oleh keterbatasan resolusi dan kemampuan penyesuaian cahaya yang kurang optimal. Pengaturan seperti *hue, saturation*, dan *gain* juga mempengaruhi kualitas gambar yang dihasilkan.

Gambar 4.11 Ukuran Lux *Led On*

Gambar 4.10 Ukuran Lux *Led Off*

Gambar 4.9 Ukuran Lux dalam ruangan mesin

Intensitas cahaya di dalam ruangan dan pada konveyor dengan lampu *LED* menyala dan mati berbeda secara signifikan. Pada Gambar 4.8, lux meter menunjukkan intensitas cahaya sebesar 1243 lux dengan suhu 34,6°C. Gambar 4.9, lux meter menunjukkan pembacaan sebesar 1238 lux saat lampu *LED* nyala pada konveyor, sementara pada Gambar 4.10, saat lampu *LED* mati, intensitas cahaya turun menjadi 262 lux. Perbedaan ini menunjukkan selisih yang cukup besar dalam intensitas cahaya di dalam ruangan, dan pentingnya menggunakan lampu *LED* untuk menyesuaikan dengan tingkat pencahayaan yang optimal. Pencahayaan yang

baik sangat mempengaruhi kualitas gambar dan pengaturan kamera, terutama dalam hal kecerahan dan kejernihan.

Kecepatan konveyor merupakan faktor yang mempengaruhi citra hasil akhir. Dari hasil observasi lapangan pada mesin konveyor, pengaturan kecepatan ternyata statis atau tetap, dan tidak menunjukkan perbedaan yang signifikan. Namun, untuk mengetahui kecepatan secara detail, dilakukan perhitungan manual bersama operator lapangan. Kecepatan konveyor tanpa beban atau tanpa adanya kulit di atasnya adalah sekitar 37,60 putaran per detik, yang kemudian berkurang menjadi 33,50 putaran per detik saat ada beban kulit di atasnya. Kecepatan ini mempengaruhi ketepatan dan konsistensi gambar yang diambil, terutama dalam proses *stitching* di mana gambar-gambar diambil dalam urutan yang cepat dan perlu digabungkan dengan tepat.

4.5 Pembahasan

Dalam proses *stitching* gambar kulit pada konveyor yang berjalan, masih ditemukan beberapa kendala. Masalah ini dimulai dari cara pengambilan gambar kulit hingga *output* akhir yang dihasilkan. Warna, pola, pencahayaan, dan titik piksel yang dipilih belum merata pada setiap *frame* yang diambil. Hal ini disebabkan oleh kurangnya titik homografi atau kecocokan keypoint yang diperlukan untuk menyatukan gambar dengan sempurna.

Pengambilan data hanya bisa dilakukan dalam format video, dan *frame* diambil berdasarkan fps yang ditentukan. Pemilihan *frame* hanya pada titik-titik tertentu dilakukan untuk mengurangi waktu yang dibutuhkan dalam proses stitching. Namun, hal ini juga dapat menyebabkan keterbatasan dalam memperoleh gambar yang lengkap dan detail. Perbedaan performa waktu pada hasil akhir disebabkan oleh variasi ukuran file video dan kualitas gambar yang diambil. Ukuran file yang lebih besar dan kualitas gambar yang lebih tinggi memerlukan lebih banyak waktu untuk diproses.

Penggunaan *stitching* vertikal dalam penelitian ini belum menghasilkan hasil yang maksimal karena pengaturan seperti nilai fps, *overlay*, dan *blending* belum sepenuhnya optimal dan konsisten untuk semua jenis citra kulit yang diambil. Namun, untuk mencapai hasil yang rata-rata maksimal, beberapa pengaturan dapat diterapkan, seperti menggunakan fps 55 atau 54 jika fps video asli adalah 116,44,

untuk mendapatkan jumlah *frame* yang memadai dan maksimal. Nilai overlay yang optimal berada di kisaran 200-350, dan nilai *blending* yang baik berada di kisaran 2.0 - 3.0. Meskipun demikian, nilai-nilai ini tidak selalu konsisten dan dapat bervariasi tergantung pada data yang diambil, sehingga masih belum pasti dalam memberikan informasi yang akurat.

Berikut proses tahapan *output stitching* berdasarkan proses yang diambil tiap kamera dari citra kulit domba, kambing, dan sapi:

1. Tahapan Stitching

   Pada tahap ini, citra-citra individual yang diambil dari perekaman yang diarahkan pada tepi objek kulit digabungkan satu per satu. Hasilnya adalah tampilan gambar yang masih terlihat terpotong-potong dengan garis-garis jelas yang menunjukkan batasan antara citra yang digabungkan dan ditambahkan dengan fungsi "remove_black_borders(image)" untuk menghapus frame hitam pada citra yang disusun.

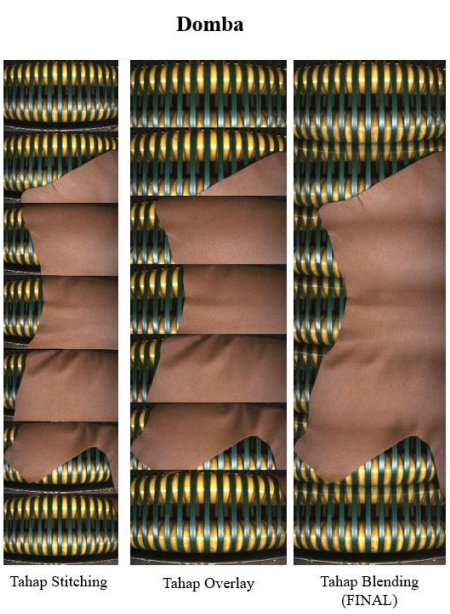
2. Tahapan Overlay

   Pada tahap overlay, citra-citra yang sudah digabungkan pada tahap sebelumnya disusun kembali dengan menyesuaikan posisi dan lapisan setiap bagian citra. Meskipun masih terlihat beberapa garis batas, namun pergeseran posisi sudah mulai diatur untuk mengurangi perbedaan yang tajam antara setiap citra mengacu pada fungsi "blend_images(img1, img2, overlap, strength=1.0)" dengan mengubah nilai overlap pada fungsi "blend_images".

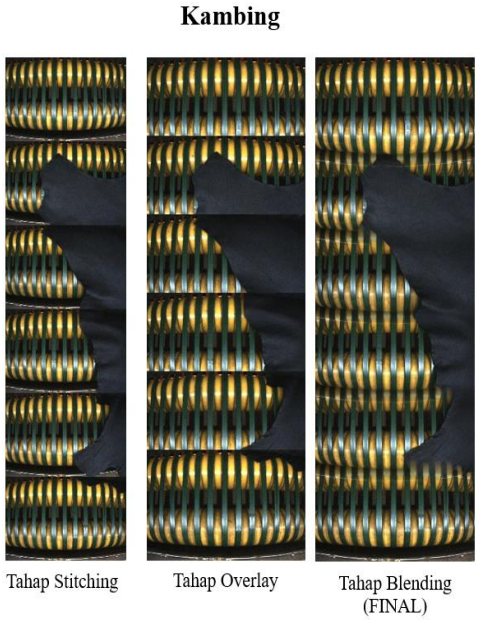
3. Tahapan Blending

   Pada tahap akhir ini, citra-citra yang telah di-overlay kemudian dibaurkan sehingga perbedaan garis batas antar-citra menjadi lebih halus. Dengan teknik blending digunakan untuk menyamarkan perbedaan warna dan tekstur, sehingga hasil akhirnya adalah gambar yang tampak lebih utuh dan mulus dengan menggunakan fungsi "blended_image = np.vstack((img1[:-overlap], blended_roi, img2[overlap:]))" untuk digabungkan area yang ditumpang tindah diatas frame kemudian dihaluskan dengan *frame* berikutnya.

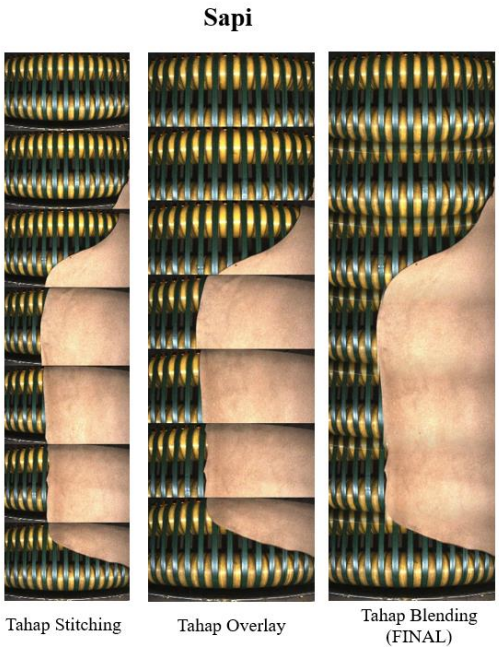
Blackfly



Gambar 4.12 Tahapan proses kulit domba dari Blackfly

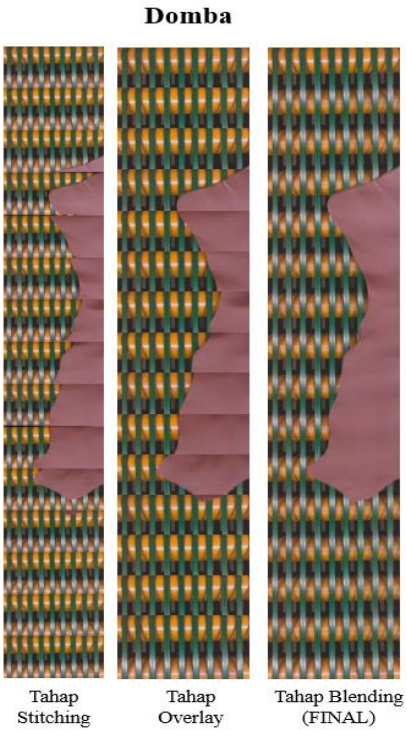


Gambar 4.13 Tahapan proses kulit kambing dari Blackfly

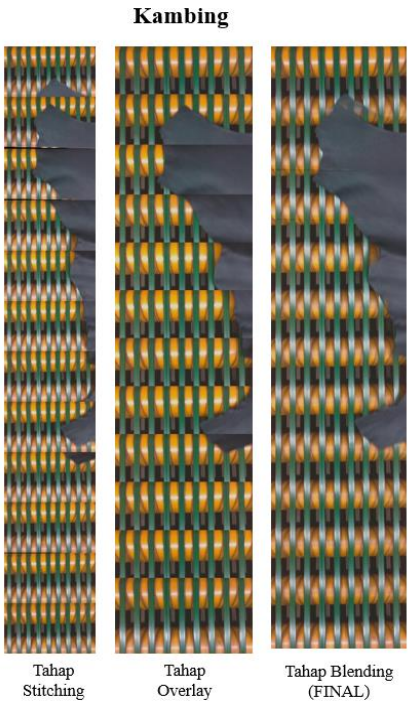


Gambar 4.14 Tahapan proses kulit sapi dari Blackfly

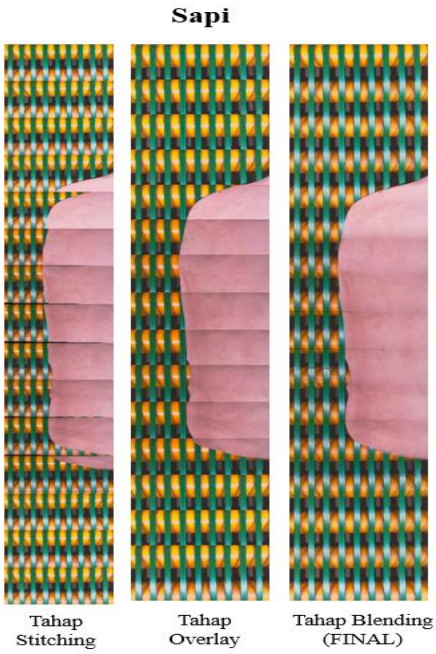
IntelRealsense



Gambar 4.15 Tahapan proses kulit domba dari IntelRealsense

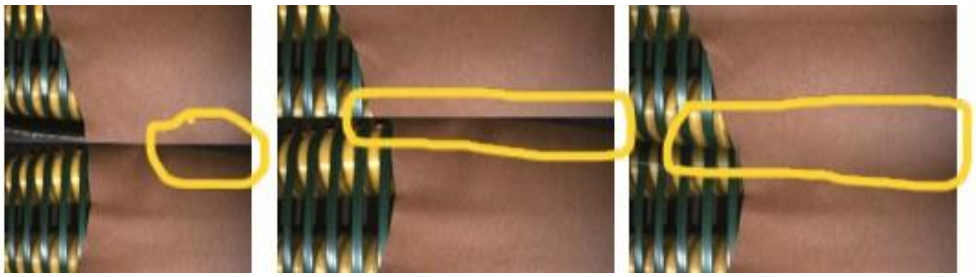


Gambar 4.16 Tahapan proses kulit kambing dari IntelRealsense



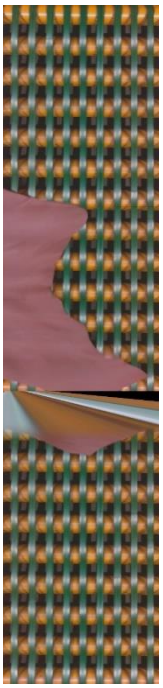
Gambar 4.17 Tahapan proses kulit sapi dari IntelRealsense

Untuk hasil saat proses *overlay*, dan *blending* tidak mengganggu atau menghilangkan *defect* pada objek kulit yang di tampilkan karena hanya mengubah atau menyambungkan *frame* detail atas dan bawahnya saja seperti gambar 4.18.

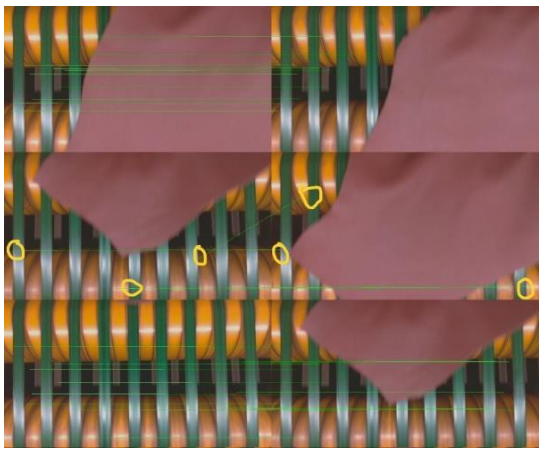


Gambar 4.18 Detail proses stitching

Kemudian jika proses stitching gagal disebabkan karena tidak cukup *keypoint* yang cocok antara kedua *frame*, dan juga perbedaan warna atau piksel antara kedua *frame*, seperti pada gambar 4.19. Dalam kasus ini, ketidakcukupan *keypoints* yang cocok dapat menghalangi perhitungan homografi yang akurat antara *frame* A dan B, mengakibatkan hasil *stitching* yang tidak sempurna dengan detail homografi yang terlalu sedikit antara kedua *frame* terlihat pada gambar 4.20.



Gambar 4.19 Hasil *stitching* gagal pada kulit domba



Gambar 4.20 Hasil pencocokan keypoint pada kulit domba

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam penelitian ini, metode stitching vertikal berhasil diterapkan untuk menyusun citra kulit yang utuh, di mana kamera Blackfly S dengan pengambilan video pada settingan 116 fps dan resolusi 1920x1200 menghasilkan gambar dengan resolusi tinggi dan detail yang lebih baik, menjadikannya ideal untuk aplikasi yang membutuhkan gambar berkualitas tinggi. Meskipun demikian, proses stitching dengan kamera ini memerlukan waktu yang lebih lama, rata-rata 11.34 detik, dibandingkan dengan kamera IntelRealsense yang menggunakan settingan 60 fps dengan resolusi 960x540. Kamera IntelRealsense, meskipun menghasilkan gambar dengan resolusi lebih rendah, namun lebih efisien dalam hal kecepatan pemrosesan, dengan waktu rata-rata 5.57 detik. Oleh karena itu, pemilihan kamera dan kecepatan konveyor harus disesuaikan dengan kebutuhan operasional perusahaan, apakah lebih memprioritaskan kualitas citra atau kecepatan pemrosesan untuk menjaga kualitas citra *output* dan kecepatan pemrosesan yang optimal sesuai dengan standar perusahaan.

Kamera Blackfly S mampu menghasilkan gambar berkualitas tinggi dengan tekstur yang lebih halus dan detail yang lebih tajam dengan besaran resolusi rata-rata keseluruhan citra 1920 x 6037 atau setara dengan 11.591.040 piksel. Hal ini membuatnya cocok untuk aplikasi yang membutuhkan resolusi tinggi, meskipun dengan respon kinerja yang relatif lambat. Sebaliknya, kamera IntelRealsense unggul dalam hal kecepatan pemrosesan, menjadikannya pilihan yang lebih baik untuk aplikasi yang memerlukan respons cepat, meskipun dengan resolusi yang kurang cukup baik dan detail ketajaman gambar dengan besaran resolusi rata-rata 960 x 5031 atau 4.829.760 piksel. Oleh karena itu, pemilihan kamera harus disesuaikan dengan kebutuhan spesifik, apakah lebih memprioritaskan kualitas gambar atau kecepatan pemrosesan.

5.2 Saran

Berdasarkan hasil percobaan dan pengujian, ada beberapa saran yang dapat digunakan untuk meningkatkan dan mengembangkan penelitian ini, di antaranya adalah sebagai berikut:

1. Penggunaan Kamera Blackfly S High Frame Rate dan IntelRealsense D435 Depth sebaiknya membutuhkan alat dan kebutuhan mesin konveyor agar tempat dan waktu pengambilan dapat berjalan dengan baik.
2. Pengujian menggunakan teknik *Deep learning,* karena sebuah pengenalan citra baru dan banyaknya jenis dibutuhkan data *train* dan data *training* untuk menghasilkan data pengenalan citra lebih maksimal.
3. Peningkatan kualitas hasil citra, disarankan menggunakan metode *pre-processing* agar hasilnya konsisten dan optimal serta meningkatkan resolusi kamera. Hal ini dapat membantu mengurangi *noise* dan meningkatkan kejelasan detail pada gambar yang diambil.
4. Tingkatkan dengan kalibrasi yang lebih akurat pada kamera yang digunakan dapat meningkatkan akurasi pengukuran dan hasil gambar. Kalibrasi ini penting terutama ketika menggunakan lebih dari satu kamera untuk memastikan bahwa semua kamera memiliki pengaturan yang seragam dan data yang dihasilkan konsisten.
5. Pengenalan citra yang bervariasi, dibutuhkan *dataset* yang digunakan untuk melatih dapat membantu model mengenali lebih banyak variasi citra, sehingga meningkatkan kemampuan generalisasi dan akurasi prediksi.
6. Mengoptimalkan pengenalan citra secara *realtime*, dibutuhkan deteksi kecacatan kulit otomatis pada mesin konveyor berjalan secara otomatis. Karena untuk proses saat ini penilaian kulit masih bergantung dalam pemeriksaan manual pada potongan kulit lebar (lebarnya sekitar 1,5 meter).
7. Peningkatan hasil *stitching* kulit menjadi lebih utuh, dibutuhkan beberapa kamera yang mempunyai detail, warna, ketajaman dan fokus yang tinggi agar penggabungan gambar secara luas dan besar.

DAFTAR PUSTAKA

Adel, E., & Elmogy, M. (2014). *Image Stitching based on Feature Extraction Techniques : A Survey Image Stitching based on Feature Extraction Techniques : A Survey*. *November*. https://doi.org/10.5120/17374-7818

Alfath Daryl Alhajir, Yisti Vita Via, & Wahyu Syaifullah Jauharis Saputra. (2021). Sistem Pendeteksi Objek Beras Dan Benda Asing Berbasis Keras Dan Google Colab. *Jurnal Informatika Dan Sistem Informasi*, *2*(3), 580–586. https://doi.org/10.33005/jifosi.v2i3.369

Beis, J. S., & Lowe, D. G. (n.d.). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1000–1006. https://doi.org/10.1109/CVPR.1997.609451

Boda, S. (2009). *Feature-based image registration*.

Boesch, G. (n.d.). *What is OpenCV? The Complete Guide (2024)*. Retrieved March 5, 2024, from https://viso.ai/computer-vision/opencv/

Fan, J. C., & Liu, Y. Z. (2015). Research on Natural Interactive 3D Registration Algorithm Based on the Detection of Human Hand Features. *Proceedings of the 2015 International Conference on Artificial Intelligence and Industrial Engineering*, *123*(Aiie), 4–7. https://doi.org/10.2991/aiie-15.2015.2

Intel® RealSense™ Depth Camera D435. (n.d.). Retrieved March 5, 2024, from https://www.intelrealsense.com/depth-camera-d435/

Jain, & Petrou. (2017). Drawback of Thesholding. *Thresholding*, 1–19. https://www.cse.unr.edu/~bebis/CS791E/Notes/Thresholding.pdf

Kurnia, D. A., Hayati, U., Hartati, T., Manikari, S. L., & Afandi, F. (2022). Pengenalan Wajah menggunakan Principle Component Analysis (PCA) dengan Model Algoritma Machine Learning untuk Mengidentifikasi Jenis Kelamin pada Kartu Identitas Mahasiswa. *TEMATIK*, *9*(2), 219–224. https://doi.org/10.38204/tematik.v9i2.1029

Lifshits, Y. (2010). Nearest neighbor search. *SIGSPATIAL Special*, *2*(2), 12–15. https://doi.org/10.1145/1862413.1862417

Liu, W., Zhang, K., Zhang, Y., He, J., & Sun, B. (2023). Utilization of Merge-Sorting Method to Improve Stitching Efficiency in Multi-Scene Image Stitching. *Applied Sciences*, *13*(5), 2791. https://doi.org/10.3390/app13052791

Manin, J., Skeen, S. A., & Pickett, L. M. (2018). Performance comparison of state-of-the-art high-speed video cameras for scientific applications. *Optical Engineering*, *57*(12), 1. https://doi.org/10.1117/1.oe.57.12.124105

Mohammadi, F. S., Mohammadi, S. E., Mojarad Adi, P., Mirkarimi, S. M. A., & Shabani, H. (2024). A comparative analysis of pairwise image stitching techniques for microscopy images. *Scientific Reports*, *14*(1), 9215. https://doi.org/10.1038/s41598-024-59626-y

Mulyawan, H., Samsono, M. Z. H., & Setiawardhana. (2011). Identifikasi Dan

Tracking Objek Berbasis Image. *Identifikas Dan Tracking Objek Berbasis Image Processing Secara Real Time*, 1–5. http://repo.pens.ac.id/1324/1/Paper_TA_MBAH.pdf

Neal, J., Leipold, T., & Petroskey, K. (2024, April 9). *The Effect of Image Stabilization on PhotoModeler Project Accuracy*. https://doi.org/10.4271/2024-01-2474

Oktaviano, R., Ripanti, E. F., & Pratiwi, H. S. (2021). Implementasi Image Stitching pada Aplikasi Virtual Tour Bandar Udara Internasional Supadio. *Jurnal Sistem Dan Teknologi Informasi (Justin)*, *9*(3), 381. https://doi.org/10.26418/justin.v9i3.45056

Ramadhan, A. W., Aulia, F., Dewi, N. M. L. A., Winarno, I., & Sukaridhoto, S. (2024). Distributed Aerial Image Stitching on Multiple Processors using Message Passing Interface. *JOIV : International Journal on Informatics Visualization*, *8*(1), 409. https://doi.org/10.62527/joiv.8.1.1890

RD. Kusmanto, A. N. T. (2011). Pengolahan Citra Digital Untuk Mendeteksi Obyek Menggunakan Pengolahan Warna Model Normalisasi Rgb. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011 (Semantik 2011*. https://doi.org/10.1016/S0166-1116(08)71924-1

Romdhoni, N. F., Usman, K., & Hidayat, B. (2020). Deteksi Kualitas Kacang Kedelai Melalui Pengolahan Citra Digital dengan Metode Gray-Level Co-Occurrence Matrix (Glcm) dan Klasifikasi Desicion Tree. *Prosiding Seminar Nasional Riset Dan Information Science (SENARIS)*, *2*, 132–137.

Rosebrock, A. (2018). *Image Stitching with OpenCV and Python*. 17 December 2018. https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/

Saputra, C. (2023). Implementasi Algoritma SIFT (Scale-Invariant Feature Transform) Dan Algoritma Kalman Filter Dalam Mendeteksi Objek Bola. *Jurnal processor*, *18*(1). https://doi.org/10.33998/processor.2023.18.1.791

Sari, D. P., Rasyad, S., Evelina, E., & Amperawan, A. (2017). Identifikasi Huruf Braille Berbasis Image Processing Secara Real Time. *Jurnal Ampere*, *2*(2), 68. https://doi.org/10.31851/ampere.v2i2.1765

Sharma, A. (2018). Decision tree implementation using Python. *GeeksforGeeks*, *2 January 2023*. https://www.geeksforgeeks.org/decision-tree-implementation-python/

Sutjipto, S. S. U., Wirayuda, T. A. B., & Wibowo, A. T. (2010). *Penggabungan Citra Panorama Secara Otomatis Menggunakan Invariant Feature*.

Tabatabaei, F., Azarmi, S., Abbaszadeh Afshar, M. J., Yarizadeh, H., & Mohtasebi, S. (2020). Blackfly fever and dermatitis caused by Simulium kiritshenkoi: a human case report in Iran. *BMC Infectious Diseases*, *20*(1), 348. https://doi.org/10.1186/s12879-020-05070-y

Yan, M., Qin, D., Zhang, G., Tang, H., & Ma, L. (2023). Nighttime Image Stitching Method Based on Image Decomposition Enhancement. *Entropy*, *25*(9), 1282. https://doi.org/10.3390/e25091282

Zhang, H., Zheng, R., Zhang, W., Shao, J., & Miao, J. (2023). An Improved SIFT Underwater Image Stitching Method. *Applied Sciences*, *13*(22), 12251. https://doi.org/10.3390/app132212251

LAMPIRAN

Stitching.py

```
import cv2
import numpy as np
import os
from panorama import Stitcher
from tkinter import Tk, filedialog

# Function to extract frames from a video
def extract_frames(video_path, frame_rate=55):
    print("Proses: Ekstraksi frame dari video dimulai")
    cap = cv2.VideoCapture(video_path)
    frames = []
    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        if frame_count % frame_rate == 0:
            frames.append(frame)
        frame_count += 1
    cap.release()
    print("Proses: Ekstraksi frame selesai, total frame yang diambil:", len(frames))
    return frames

# Function to open a file dialog and get the video path
def get_video_path():
    root = Tk()
    root.withdraw()  # Hide the root window
        video_path = filedialog.askopenfilename(title="Masukkan file video",
filetypes=[("Video files", "*.mp4;*.avi;*.mov")])
    if not video_path:
```

```
        raise ValueError("Mohon masukkan file video dengan benar")
    print("Proses: Video dipilih -", video_path)
    return video_path

# Function to open a directory dialog and get the directory path
def get_directory_path():
    root = Tk()
    root.withdraw()  # Hide the root window
     directory_path = filedialog.askdirectory(title="Pilih folder untuk menyimpan gambar")
    if not directory_path:
        raise ValueError("Pilih file direktori dengan benar!")
    print("Proses: Direktori untuk menyimpan gambar dipilih -", directory_path)
    return directory_path

# Function to save images
def save_image(image, directory, filename):
    if not os.path.exists(directory):
        os.makedirs(directory)
    cv2.imwrite(os.path.join(directory, filename), image)
    print(f"Proses: Gambar disimpan sebagai {filename} di {directory}")

# Function to remove black borders
def remove_black_borders(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    _, thresh = cv2.threshold(gray, 1, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        x, y, w, h = cv2.boundingRect(contours[0])
        image = image[y:y+h, x:x+w]
    return image
```

```
# Function to blend images
def blend_images(img1, img2, overlap, strength=1.0):
    print("Proses: Blending gambar")
      if img1.shape[1] != img2.shape[1]: #baris code untuk file gambar yang dicompress
        width = min(img1.shape[1], img2.shape[1])
        img1 = cv2.resize(img1, (width, img1.shape[0]))
        img2 = cv2.resize(img2, (width, img2.shape[0]))

    # if img1.shape[1] != img2.shape[1]: #aktifkan ini untuk tidak dicompress
    #     raise ValueError("Lebar gambar harus sama untuk blending yang benar.")

    h1, w1 = img1.shape[:2]
    h2, w2 = img2.shape[:2]

    roi1 = img1[-overlap:]
    roi2 = img2[:overlap]

    alpha = np.linspace(0, 1, overlap) ** strength
    alpha = np.tile(alpha, (w1, 1)).T

    blended_roi = roi1 * (1 - alpha)[:, :, None] + roi2 * alpha[:, :, None]

    blended_image = np.vstack((img1[:-overlap], blended_roi, img2[overlap:]))

    return blended_image

# Function to apply GrabCut to an image
def apply_grabcut(image_path):
    print("Proses: Mengaplikasikan GrabCut")
    image_real = cv2.imread(image_path)
```

```
    if image_real is None:
        raise ValueError("Error: Gambar tidak dapat dimuat. Pastikan file path sudah benar.")

    # Convert to grayscale
    gray = cv2.cvtColor(image_real, cv2.COLOR_BGR2GRAY)

    # Apply GaussianBlur to reduce noise and improve edge detection
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)

    # Adjust the threshold values for Canny edge detection
    cv2.Canny(blurred, 30, 100)  # Lower and upper threshold values

    # Use the GrabCut algorithm for better segmentation
    mask_gc = np.zeros(image_real.shape[:2], np.uint8)
    bgdModel = np.zeros((1, 65), np.float64)
    fgdModel = np.zeros((1, 65), np.float64)

    # Define the initial bounding box for the GrabCut algorithm
    rect = (50, 50, image_real.shape[1] - 50, image_real.shape[0] - 50)

    # Apply the GrabCut algorithm
        cv2.grabCut(image_real, mask_gc, rect, bgdModel, fgdModel, 30, cv2.GC_INIT_WITH_RECT) #ukuran iterasi "30-+" untuk penyempurnaan segmentasi pada algoritma grabcut

    # Modify the mask
    mask_gc2 = np.where((mask_gc == 2) | (mask_gc == 0), 0, 1).astype('uint8')

    # Refine mask using morphology operations
    kernel = np.ones((5, 5), np.uint8)
```

```
    mask_gc2 = cv2.morphologyEx(mask_gc2, cv2.MORPH_CLOSE, kernel, iterations=3)
    mask_gc2 = cv2.morphologyEx(mask_gc2, cv2.MORPH_OPEN, kernel, iterations=2)

  # Additional erosion to remove small unwanted pixels
   mask_gc2 = cv2.erode(mask_gc2, kernel, iterations=20) #Ukuran iterasi erosi "20-+" untuk menghilangkan noise atau piksel kecil yang tidak digunakan

  # Apply bitwise_and to combine original image with mask
  grabcut_output = cv2.bitwise_and(image_real, image_real, mask=mask_gc2)

  print("Proses: GrabCut selesai.")
  return grabcut_output

# Get video path using tkinter dialog
video_path = get_video_path()

# Extract frames from video
frames = extract_frames(video_path, frame_rate=9)  # UKURAN FPS VIDEO tergantung fps video (Intel: 15-+ // blackfly : 55-+ //Nikon kamera D7500 : 9-+ fps)
if len(frames) < 2:
  raise ValueError("Tidak cukup frame untuk digabungkan")

# Stitch frames together vertically without resizing
stitcher = Stitcher()
stitched_images = []
keypoint_matches = []

i = 0
while i < len(frames) - 1:
  frameA = frames[i]
```

```
    frameB = frames[i + 1]
    print(f"Proses: Stitching frame {i} dan {i+1}")
    result = stitcher.stitch([frameA, frameB], showMatches=True)
    if result is not None:
        (stitched, vis) = result
        if stitched is not None:
            stitched = remove_black_borders(stitched)
            stitched_images.append(stitched)
            if vis is not None:
                keypoint_matches.append(vis)
            i += 1  # Proceed to the next frame
        else:
            print(f"Proses: Matriks homografi tidak valid untuk frame {i} dan {i+1}, melewati frame ini.")
            i += 1  # Proceed to the next frame
    else:
            print(f"Proses: Tidak cukup keypoints yang cocok untuk menghitung homografi antara frame {i} dan {i+1}, melewati frame ini.")
        i += 1  # Skip to the next frame

total_keypoints = stitcher.getTotalKeypoints()
print(f"Jumlah total keypoints dari semua frame: {total_keypoints}")

if len(stitched_images) == 0:
    raise ValueError("Stitching gagal.")

# Add overlap between stitched images and blend them
overlap = 500  # UKURAN OVERLAP (intel: 200-+ // blackfly: 350-+ //Camera DSLR Nikon : 500-+)
blending_strength = 2.0  # UKURAN BLENDING (Min 2.0 - 5.0 Maks)
vertical_panorama = stitched_images[0]
```

```
for img in stitched_images[1:]:
        vertical_panorama = blend_images(vertical_panorama, img, overlap, strength=blending_strength)

# Create a single vertical image for keypoint matches
if len(keypoint_matches) > 0:
    vertical_keypoints = keypoint_matches[0]
    for kp in keypoint_matches[1:]:
        vertical_keypoints = cv2.vconcat([vertical_keypoints, kp])
else:
    vertical_keypoints = None

# Show the final vertical
# Show the final vertical panorama and keypoint matches
cv2.imshow("Vertical Panorama", vertical_panorama)
if vertical_keypoints is not None:
    cv2.imshow("Vertical Keypoint Matches", vertical_keypoints)

# Get directory path using tkinter dialog
save_directory = get_directory_path()

# Save the images
panorama_path = os.path.join(save_directory, "vertical_panorama.jpg")
save_image(vertical_panorama, save_directory, "vertical_panorama.jpg")
if vertical_keypoints is not None:
    save_image(vertical_keypoints, save_directory, "vertical_keypoints.jpg")

# Ask user if they want to apply GrabCut to the stitched image
apply_grabcut_option = input("Apakah Anda ingin menerapkan GrabCut pada gambar panorama yang disimpan? (y/n): ")

if apply_grabcut_option.lower() == 'y':
```

```
    # Get image path using tkinter dialog
    grabcut_image_path = filedialog.askopenfilename(title="Pilih gambar panorama untuk GrabCut", initialdir=save_directory, filetypes=[("Image files", "*.jpg;*.jpeg;*.png")])
    if grabcut_image_path:
        grabcut_output_save = apply_grabcut(grabcut_image_path)
                                    save_image(grabcut_output_save, save_directory, "vertical_panorama_removebg.jpg")

cv2.waitKey(0)
cv2.destroyAllWindows()
```

panorama.py

```
import numpy as np
import imutils
import cv2

class Stitcher:
    def __init__(self):
        self.isv3 = imutils.is_cv3(or_better=True)
        self.total_keypoints = 0

    def stitch(self, images, ratio=0.75, reprojThresh=4.0, showMatches=False):
        (imageB, imageA) = images
        (kpsA, featuresA) = self.detectAndDescribe(imageA)
        (kpsB, featuresB) = self.detectAndDescribe(imageB)

            M = self.matchKeypoints(kpsA, kpsB, featuresA, featuresB, ratio, reprojThresh)
        if M is None:
            print("Proses: Tidak cukup keypoints yang cocok untuk menghitung homografi.")
```

```
            return None

        (matches, H, status) = M
        if H is None:
            print("Proses: Matriks homografi tidak valid.")
            return None

        try:
                    result = cv2.warpPerspective(imageA, H, (imageA.shape[1],
imageA.shape[0] + imageB.shape[0]))
            result[0:imageB.shape[0], 0:imageB.shape[1]] = imageB
        except cv2.error as e:
            print(f"Proses: Kesalahan saat menggunakan warpPerspective: {e}")
            return None

        if showMatches:
            vis = self.drawMatches(imageA, imageB, kpsA, kpsB, matches, status)
            return (result, vis)

        return (result, None)

    def detectAndDescribe(self, image):
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        if self.isv3:
            descriptor = cv2.SIFT_create()
            (kps, features) = descriptor.detectAndCompute(image, None)
        else:
            detector = cv2.FeatureDetector_create("SIFT")
            kps = detector.detect(gray)
            extractor = cv2.DescriptorExtractor_create("SIFT")
            (kps, features) = extractor.compute(gray, kps)
```

```
    kps = np.float32([kp.pt for kp in kps])
    num_keypoints = len(kps)
    self.total_keypoints += num_keypoints
    print(f"Jumlah total keypoint yang terdeteksi: {len(kps)}")
    return kps, features

def getTotalKeypoints(self):
    return self.total_keypoints

def matchKeypoints(self, kpsA, kpsB, featuresA, featuresB, ratio, reprojThresh):
    matcher = cv2.DescriptorMatcher_create("BruteForce")
    rawMatches = matcher.knnMatch(featuresA, featuresB, 2)
    matches = []

    for m in rawMatches:
        if len(m) == 2 and m[0].distance < m[1].distance * ratio:
            matches.append((m[0].trainIdx, m[0].queryIdx))

    if len(matches) > 4:
        ptsA = np.float32([kpsA[i] for (_, i) in matches])
        ptsB = np.float32([kpsB[i] for (i, _) in matches])
        H, status = cv2.findHomography(ptsA, ptsB, cv2.RANSAC, reprojThresh)
        return (matches, H, status)

    return None

def drawMatches(self, imageA, imageB, kpsA, kpsB, matches, status):
    (hA, wA) = imageA.shape[:2]
    (hB, wB) = imageB.shape[:2]
    vis = np.zeros((max(hA, hB), wA + wB, 3), dtype="uint8")
    vis[0:hA, 0:wA] = imageA
```

```
vis[0:hB, wA:] = imageB

for ((trainIdx, queryIdx), s) in zip(matches, status):
   if s == 1:
      ptA = (int(kpsA[queryIdx][0]), int(kpsA[queryIdx][1]))
      ptB = (int(kpsB[trainIdx][0]) + wA, int(kpsB[trainIdx][1]))
      cv2.line(vis, ptA, ptB, (0, 255, 0), 1)
return vis
```