PROYEK AKHIR - VE180626

PERANCANGAN SISTEM STABILISASI VIDEO SEBAGAI FITUR PENDUKUNG PADA KAMERA UAV

Rachma Aji Solicha
10311600000006

Pembimbing
Imam Arifin, S.T., M.T.
Prof. Lam Siew Kei
Dimas Agus Fahrudin, S.Si.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

FINAL PROJECT – VE180626

DESIGN OF VIDEO STABILIZATION SYSTEM AS A SUPPORT FEATURE IN THE UAV CAMERA

Rachma Aji Solicha
10311600000006

Supervisor
Imam Arifin, S.T., M.T.
Prof. Lam Siew Kei
Dimas Agus Fahrudin, S.Si.

ELECTRICAL AUTOMATION ENGINEERING DEPARTMENT
Vocational Faculty
Institute of Technology Sepuluh Nopember
Surabaya 2019

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN PROYEK AKHIR

Dengan ini kami menyatakan bahwa isi sebagian maupun kesuluruhan Proyek Akhir ini dengan judul:

"PERANCANGAN SISTEM STABILISASI VIDEO SEBAGAI FITUR PENDUKUNG PADA KAMERA UAV"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang diakui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ditemukan pelanggaran, penulis bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 24 Juli 2019

Rachma Aji Solicha
NRP.10311600000006

Halaman ini sengaja dikosongkan

PERANCANGAN SISTEM STABILISASI VIDEO SEBAGAI FITUR PENDUKUNG PADA KAMERA UAV

PROYEK AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Pembimbing 1	Pembimbing 2
Imam Arifin, S.T., M.T.	Prof. Ir. Tse Kai Kei
NIP. 19730222 200212 1 001	

SURABAYA
JULI, 2019

Halaman ini sengaja dikosongkan

PERANCANGAN SISTEM STABILISASI VIDEO SEBAGAI FITUR PENDUKUNG PADA KAMERA UAV DI PT BHIMASENA RESEARCH AND DEVELOPMENT

PROYEK AKHIR

Disusun oleh:

Rachma Aji Solicha NRP. 10311600000006

Menyetujui,

Kepala *Human Resources Department*, Pembimbing Perusahaan,

BHIMASENA Research & Development

Muhammad Nur Cahyo Utomo
NIK. 021042017

Dimas Agus Fahrudin, S.Si.
NIK. 020562016

Halaman ini sengaja dikosongkan

PERANCANGAN SISTEM STABILISASI VIDEO SEBAGAI FITUR PENDUKUNG PADA KAMERA UAV

Rachma Aji Solicha
10311600000006

Pembimbing I : Imam Arifin, S.T.,M.T.
Pembimbing II : Prof. Lam Siew Kei
Pembimbing III : Dimas Agus Fahrudin S.Si.

ABSTRAK

Video yang direkam pada UAV seringkali mengalami efek getaran karena gerakan kamera yang tidak diinginkan. Kondisi video yang tidak stabil dapat mempersulit pengguna dalam mengamati target tertentu. Oleh karena itu, pada penelitian ini dirancang sebuah sistem stabilisasi video berbasis *digital image processing* untuk meredam efek getaran dalam video.

Sistem stabilisasi video terdiri dari empat proses utama. Pertama, proses deteksi dan pelacakan *feature point* antara dua *frame* video yang berurutan untuk mendapatkan jalur pergerakan kamera. Proses ini menerapkan metode Shi-Tomasi *corner detector* dan *Optical flow matching*. Kedua, proses estimasi gerak dengan pendekatan *affine transform*. Ketiga, konsep *simple low pass filter* diterapkan untuk menentukan nilai kompensasi pergerakan objek dalam video. Terakhir, data *frame* video ditransformasi berdasarkan nilai kompensasi untuk meredam efek gerak getar.

Hasil pengujian terhadap penerapan sistem stabilisasi video menunjukkan peningkatan kualitas video dengan spesifikasi yakni penurunan level SAD sebesar 1,59 dan MSE sebesar 176,11, serta peningkatan level SSIM sebesar 0,019. Kecepatan proses pengolahan gambar mencapai 21 FPS (*frame per second*).

Kata Kunci: *Unmanned Aerial Vehicle* (UAV), *Feature point,* Shi-Tomasi *corner detector, Optical flow matching, Simple low pass filter*

Halaman ini sengaja dikosongkan

DESIGN OF VIDEO STABILIZATION SYSTEM AS A SUPPORT FEATURE IN UAV CAMERAS

Rachma Aji Solicha
10311600000006

Supervisor I : Imam Arifin, S.T.,M.T.
Supervisor II : Prof. Lam Siew Kei
Supervisor III : Dimas Agus Fahrudin, S.Si.

ABSTRACT

Videos recorded on UAVs often suffer vibration effects due to unwanted camera movements. Unstable videos make it difficult for users to observe certain targets. Therefore, this research designed a video stabilization system using image processing to reduce vibration effects in a video.

Video stabilization system consists of four main processes. Firstly, a feature point detection and tracking process between two consecutive frames to get the camera movement path. This process applied Shi-Tomasi corner detector and Optical flow matching. Secondly, motion estimation is performed based on affine transform. Third, the simple low pass filter is applied to determine a motion compensation for the object movement in the video. Lastly, video frames is transformed based on compensation value to reduce the vibration effect.

The test results on the implementation of the video stabilization system show video quality improvement based on a decrease in SAD level of 1,59 and MSE of 176,11, and an increase in the SSIM level of 0,019. Speed of the image processing reach 21 FPS (frame persecond).

Keywords*: Unmanned Aerial Vehicle (UAV), Feature point, Shi-Tomasi corner detector, Optical flow matching, Simple low pass filter*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan berkah, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan Proyek Akhir yang berjudul: **"Perancangan Sistem Stabilisasi Video Sebagai Fitur Pendukung pada Kamera UAV"**. Proyek Akhir ini disusun sebagai salah satu syarat kelulusan yang harus ditempuh oleh setiap mahasiswa di Departemen Teknik Elektro Otomasi Fakultas Vokasi Institut Teknologi Sepuluh Nopember, Surabaya.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada berbagai pihak yang telah memberikan bantuan dan dukungan dalam penyelesaian Proyek Akhir. Ucapan terimakasih saya tujukan pada kedua orang tua, keluarga besar saya, Bapak joko susilo selaku kepala departemen berserta seluruh dosen pengajar dan staf pengajaran, Bapak Imam Arifin selaku pembimbing beserta seluruh jajaran dosen pembimbing lainnya, Bapak Aris Budiyatno selaku CEO PT. Bhimasena Research and Development beserta seluruh rekan kerja selama magang dan seluruh anggota laboratorium CyPIRAL atas dukungan, semangat dan bimbingan yang penulis terima.

Penulis menyadari bahwa Proyek Akhir ini masih jauh dari kesempurnaan dengan segala kekurangannya. Oleh karena itu, penulis mengharapkan adanya kritik dan saran dari semua pihak demi kesempurnaan dari Proyek Akhir ini. Besar harapan penulis bahwa Proyek Akhir ini dapat memberikan informasi dan manfaat bagi pembaca pada umumnya mahasiswa Departemen Teknik Elektro Otomasi.

Surabaya, 24 Juli 2019

Penulis
Rachma Aji Solicha

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Teknologi *unmanned aerial vehicle* (UAV) seringkali digunakan untuk membantu berbagai misi dibidang pertahahan dan keamanan. Seiring dengan perkembangan teknologi, produk UAV yang ditawarkan semakin beragam. PT Bhimasena Research and Development melalui divisi *unmanned aerial vehicle-backpack* (UAV-Backpack) mengembangkan sebuah produk UAV berjenis *Fixed Wing* yang dirancang khusus untuk misi pengitaian. UAV tersebut dilengkapi dengan perangkat kamera dan sistem pengolahan data gambar. Sistem pengolahan data gambar terdiri dari kamera itu sendiri, *single board computer*, *transmitter*, dan *Ground Control Station* (GCS). Perangkat *single board computer* bertugas untuk mengakses data gambar hasil tangkapan kamera untuk kemudian dirangkai menjadi sebuah file video. Hasil pengolahan video kemudian melalui perangkat *transmitter* dikirimkan ke GCS. Perangkat GCS sendiri memberikan akses pada pengguna untuk melihat hasil pengolahan video. Kualitas video yang dihasilkan memiliki peranan penting dalam misi pengintaian target. Namun, video yang direkam pada UAV seringkali mengalami efek getaran karena gerakan kamera yang tidak diinginkan. Gerakan kamera tersebut disebabkan oleh pengaruh gesekan udara, getaran mesin hingga gerak maneuver UAV. Kondisi video yang tidak stabil dapat mempersulit pengguna dalam mendeteksi dan melacak target tertentu. Sehingga, diperlukan sebuah fitur pendukung pada perangkat kamera UAV untuk meningkatkan kualitas video yang dihasilkan.

Stabilisasi video merupakan teknologi tambahan pada video yang bertujuan untuk meminimalisir getaran dari video [1]. Secara umum ada dua jenis teknik stabilisasi yakni stabilisasi video secara mekanis dan stabilisasi video berbasis pengolahan digital. Pada teknik stabilisasi secara mekanis menggunakan perangkat eksternal seperti peredam kejut, giroskop dan lain sebagainya untuk menstabilkan tampilan video. Namun, metode tersebut kurang mampu menghilangkan gangguan berupa getaran pada kamera. Sedangkan, teknik stabilisasi video berbasis pengolahan digital menggunakan metode pelacakan

Feature Point atau titik acuan untuk mendapatkan jalur pergerakan kamera [2]. Metode ini memungkinkan sistem untuk mendeteksi serta meminimalisir getaran dalam video.

Dalam proyek akhir ini dikembangkan sebuah konsep sistem stabilisasi video sebagai fitur pendukung pada perangkat kamera UAV. Sistem stabilisasi video dirancang untuk mengolah data gambar hasil tangkapan kamera dengan cepat dan mampu meredam gangguan pada gambar yang dihasilkan. Sistem tersebut memuat keseluruhan algoritma dari metode stabilisasi video berbasis *Digital Image Processing* atau *Digital Video Stabilization.* Penambahan sistem stabilisasi video pada sistem pengolahan data gambar diharapkan mampu meningkatkan kualitas tampilan video. Selain itu, penelitian ini diharapkan dapat menjadi sumber referensi bagi pihak perusahaan untuk pengembangan produk UAV maupun teknologi stabilisasi video itu sendiri.

1.2. Rumusan Masalah

Unmanned Aerial Vehicle (UAV) seringkali digunakan untuk mengintai target tertentu. Namun, perilaku UAV menyebabkan gangguan pada tampilan video. Tampilan video yang tidak stabil dapat menyulitkan pengguna dalam melakukan pengamatan terhadap objek tertentu. Oleh karena itu, diperlukan sebuah fitur pendukung berupa sistem stabilisasi video untuk meningkatkan kualitas video pada UAV.

1.3. Batasan Masalah

Sistem stabilisasi video yang dikembangkan dirancang untuk mampu mendeteksi serta meredam gangguan berupa gerak getar pada tampilan video. Durasi pengolahan gambar tidak melebihi 100 milisekon. Perancangan sistem stabilisasi video memanfaatkan *framework* Qt.5 berbasis bahasa pemograman C++. Pengujian dilakukan menggunakan perangkat *personal computer* dengan bantuan modul uji.

1.4. Tujuan

Tujuan dari penelitian ini adalah merancang sebuah sistem stabilisasi video yang mampu meredam gangguan pada tampilan video serta memiliki durasi pengolahan data gambar yang singkat.

1.5. Metodologi

Dalam penyelesaian proyek akhir perancangan sistem stabilisasi video pada kamera UAV memerlukan beberapa tahapan. Tahapan yang

dimaksud antara lain tahap persiapan, perancangan konsep sistem, perancangan tahapan dan metode pengolahan data, pengujian dan analisa data serta penulisan laporan. Pada tahap persiapan dilakukan studi literatur mengenai konsep dari *digital image processing*, teknologi stabilisasi video, pemograman berbasis C++ serta *framework* Qt.5. Pada tahap perancangan tahapan dan metode pengolahan data, dilakukan perancangan sistem stabilisasi video yang memuat keseluruhan tahap serta metode pengolahan gambar Selanjutnya, pada tahap pengujian dilakukan perancangan modul pengujian untuk mendapatkan data hasil uji performa sistem stabilisasi video. Pengujian dilakukan menggunakan perangkat *personal computer* dan sistem *graphical user interface* (GUI). Kemudian dilakukan analisa terhadap hasil data uji yang diperoleh. Proses analisa dilakukan berdasarkan berbagai sumber referensi dari penelitian sebelumya. Pada tahap terakhir dilakukan penyusunan laporan akhir. Laporan tersebut memuat keseluruhan tahap penyelesaian proyek akhir yang dilakukan.

1.6. Sistematika Penulisan

Pada laporan Proyek Akhir ini disusun dengan sistematika sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini memuat latar belakang dari proyek akhir, rumusan dan batasan masalah yang akan diangkat, tujuan penelitian, metodologi yang digunakan serta sistematika laporan Proyek Akhir.

BAB II UNMANNED AERIAL VEHICLE (UAV) DAN TEKNIK STABILISASI VIDEO BERBASIS DIGITAL IMAGE PROCESSING

Keseluruhan pembahasan dalam bab ini melliputi *Unmanned Aerial Vehicle* (UAV), teknologi stabilisasi video, *digital image processing*, metode pengolahan data, metode pengujian performa sistem, konsep pemograman C++, dan *Framework* Qt.

BAB III PERANCANGAN SISTEM STABILISASI VIDEO DAN MODUL UJI PERFORMA

Pembahasan berfokus pada konsep sistem stabilisasi video yang dikembangkan, perancangan tahap serta metode pengolahan

data, serta perancangan modul uji yang memuat keseluruhan tahap pengujian sistem stabilisasi video.

BAB IV PENGUJIAN DAN ANALISA DATA

Bab ini menyajikan keseluruhan metode pengujian dan analisa performa sistem stabilisasi video. Data hasil pengujian performa sistem stabilisasi video dianalisa berdasarkan berbagai sumber penelitian sebelumnya.

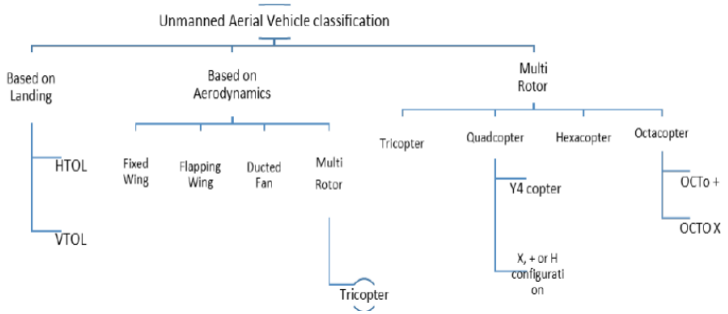
BAB V KESIMPULAN DAN SARAN

Dalam bab ini memuat kesimpulan dari hasil pengujian serta analisa data dan saran untuk penelitian berikutnya.

BAB II
UNMANNED AERIAL VEHICLE (UAV) DAN TEKNIK STABILISASI VIDEO BERBASIS DIGITAL IMAGE PROCESSING

2.1. *Unmanned Aerial Vehicle* (UAV)

Teknologi *Unmanned Aerial Vehicle* (UAV) merupakan kendaraan udara yang mampu melakukan penerbangan berkelanjutan tanpa membutuhkan operator manusia di dalam pesawat [3]. UAV dapat dioperasikan menggunakan mode *remotely controlled, semi-autonomous, autonomous* maupun kombinasi dari ketiga mode tersebut. UAV juga dirancang untuk mampu mengerjakan berbagai tugas [3]. Saat ini UAV sebagian besar digunakan dalam aplikasi militer. Selain itu, UAV juga dapat diaplikasikan dalam bidang *scientific, public safety* dan berbagai aplikasi komersial seperti pengumpulan data dan *image acquisition* dari area yang terdampak, pemetaan wilayah dan lain sebagainya [3]. Secara umum, UAV dapat dibedakan berdasarkan cara mendarat (*landing*), aerodinamik dan jenis *multi rotor* yang digunakan.



Gambar 2.1 Klasifikasi *Unmanned Aerial Vehicle* (UAV) [4]

Pada Gambar 2.1 dapat dilihat bahwa UAV memiliki beberapa jenis berdasarkan beberapa parameter. Berdasarkan struktur aerodinamik, UAV dibedakan menjadi *Fixed wing, flapping wing, ducted fan* dan *multi rotor* [4]. Salah satu jenis UAV yang umum digunakan adalah UAV jenis *fixed wing.* Struktur pesawat dapat dilihat pada Gambar 2.2 yakni sebagai berikut,

Gambar 2.2 UAV *Fixed Wing* [36]

Tantangan dalam mendesain pesawat bergantung pada jenis aplikasi yang menentukan area cakupan, ketinggian maksimum, kecepatan, tingkat pendakian, waktu penerbangan atau daya tahan, dan stabilitas [5]. Semua spesifikasi cenderung bervariasi tergantung pada aplikasi dan efek lingkungan [4]. Perancangan *unmanned aerial system* (UAS) mencakup UAV itu sendiri dan beberapa subsistem yang mencakup hubungan komunikasi antara UAV dan pengguna seperti *ground control station* (GCS) dan aksesoris seperti gimbal dan payload [4]. Berikut gambaran umum dari UAS dapat dilihat pada Gambar 2.3.

Gambar 2.3 *Unmanned Aerial System* (UAS) [37]

Komponen utama subsistem pesawat adalah unit pengukuran inersia, motor, *propellers* dan *receiver*, *processor* dan *Airframe* [4]. Bahan logam yang paling umum untuk membuat pesawat adalah paduan, aluminium dan titanium, sedangkan bahan bukan logam termasuk plastik transparan dan diperkuat [6].

2.2. *Digital Image Processing*

Konsep *digital image processing* terdiri dari beberapa teknik pengolahan data gambar digital menggunakan perangkat komputer [7]. Gambar dapat didefinisikan sebagai fugsi dua dimensi f(x,y), dengan x dan y sebagai koordinat spatial, serta nilai amplitude dari fungsi f untuk setiap koordinat disebut dengan intensitas atau *gray level* dari gambar pada titik tersebut [7]. Gambar digital memilki nilai koordinat x,y dan intensitas dari fungsi yang terbatas, serta bernilai diskrit [7]. Salah satu elemen penting dalam gambar digital adalah piksel. Piksel merupakan representasi dari element yang terdapat pada gambar digital. Elemen tersebut berupa lokasi dan nilai intensitas warna dari kumpulan titik yang membentuk gambar tersebut [7]. Seiring dengan perkembangan aplikasi luar angkasa, teknik *digital image processing* mulai berkembang pada akhir 1960 dan awal tahun 1970 untuk penggunaan di pencitraan medis, sumber daya bumi terpencil dan astronomi [7].

Dari tahun 1960 hingga saat ini, pembahasan mengenai pemrosesan gambar telah berkembang pesat [7]. Dalam penerapannya untuk aplikasi di bidang kesehatan dan program luar angkasa, teknik *digital image processing* saat ini menggunakan berbagai aplikasi. Prosedur komputer digunakan untuk meningkatkan kontras atau kode dari level intensitas menjadi warna untuk memperuda interpretasi dari X-ray dan gambar lain yang di gunakan dalam industry, kesehatan dan sains biologi [7]. Secara umum, terdapat beberapa teknik pemrosesan gambar antara lain *Image Acquisition, Image filtering and enhancement, Image restoratium, Color image processing, Wavelets and multiresolution processing, Compression, Morphological processing, Segmentation, Representation and description* dan *Object recognition*.

Berbagai proses diatas dapat diterapkan sesuai dengan kebutuhan dari sistem. Sebagian besar dari gambar dapat dikelompokkan berdasarkan kombinasi dari sumber iluminasi dan refleksi dari energi yang diperoleh dari *scene* yang ditangkap [7]. Dari berbagai jenis gambar dan teknik pemrosesan yang digunakan, kita dapat memperoleh berbagai informasi. Informasi tersebut dapat digunakan untuk menyelesaikan beberapa tugas antara lain tugas yang berbasis *computer vision*.

2.3. Stabilisasi Video

Proses stabilisasi video merupakan proses untuk menghasilkan kompensasi baru dalam urutan frame video, dimana gerakan dalam

tampilan gambar yang tidak diinginkan berupa *jitter* (getaran) dihapus [2]. Keberadaan teknologi ini digunakan untuk meningkatkan kualitas suatu video. Peningkatan kualitas video menjadi topic penting seiring dengan keberadaan *digital media visual*. *Image Stabilization* atau *Video Stabilization* bertujuan untuk menstabilkan gerakan goyang (*Shaky)* atau urutan video yang tidak stabil [8]. Secara umum, terdapat tiga metode *Video Stabilization* yakni *Mechanical Video Stabilization*, *Optical Image Stabilization* dan *Digital Image Stabilization* (DIS) [9]. Sedangkan, menurut M. Chethan Kumar [10] secara umum ada dua jenis teknik stabilisasi yakni stabilisasi video secara mekanis dan stabilisasi video berbasis pengolahan digital.

Pada teknik stabilisasi secara mekanis (*Mechanical Video Stabilization*) menggunakan perangkat eksternal seperti peredam kejut, giroskop dan lain sebagainya untuk menstabilkan tampilan video. Sedangkan, teknik stabilisasi video berbasis pengolahan digital (*Digital Video Stabilization*) menggunakan metode pelacakan *Feature Point* atau *Key Point* untuk mendapatkan jalur pergerakan kamera. Stabilisasi video umumnya terdiri dari empat tahap yakni *Image pre-processing*, *Image matching*, *Motion estimation* dan *Motion compensation* [11].

Pada tahap *Image pre-processing* dilakukan proses untuk mendeteksi keberadaan *feature point*. Tahap ini mencangkup beberapa proses pengolahan gambar untuk memperoleh data *feature point.* Proses tersebut antara lain yakni konversi gambar dalam format *grayscale image* dan *feature point detectors*. Keberadaan *feature point* berperan sebagai tolok ukur untuk mendeteksi adanya gangguan pada tampilan video. Pada tahap *Image Matching* dilakukan proses pelacakan posisi *feature point* untuk memperoleh vector gerak dari *feature point* tersebut. Vector gerak (*motion vector*) kemudian diolah pada tahap *motion estimation*.

Pada tahap ini, dilakukan estimasi atau perkiraan gerak pada tampilan video. Selanjutnya, pada tahap *motion compensation* dilakukan pemisahan gerak yang tidak diinginkan berupa getaran dan perubahan sudut pengambilan gambar pada tampilan video. Hasil akhir dari keseluruhan proses pengolahan video berupa urutan frame video yang stabil. Teknologi stabilisasi video sringkali digunakan dalam bidang kedokteran, militer dan robotika [9]. Keberadaan teknologi ini membantu manusia untuk memperoleh informasi yang lebih akurat dari sebuah gambar maupun video.

2.4. *Feature Point Detectors*

Dalam *image processing* dan beberapa aplikasi berbasis *computer vision*, data gambar yang akan diolah perlu direpresentasikan dalam bentuk sekumpulan *feature point* [1]. Hal ini dilakukan sebagai bentuk effisiensi kerja perangkat komputer. Proses *feature point* detectors merupakan proses pengolahan data gambar untuk mendeteksi keberadaan *feature point* atau *interest point* dalam sebuah gambar. *Feature point* merupakan titik piksel pada gambar yang memuat informasi tertentu dari sebuah gambar. Pada proses stabilisasi video, *feature point* digunakan sebagai tolok ukur untuk mengidentifikasi distorsi gerak (*motion distortion*) antara dua frame yang berurutan. *Feature point detector* secara umum dapat diklasifikasikan menjadi *single-scale detectors, multi-scale detectors* dan *affine invariant detectors* [1]. *Single scale detector* merupakan metode deteksi *feature point* berdasarkan parameter internal yang telah ditetapkan [1]. Terdapat beberapa jenis *feature point detectors* yakni antara lain *Scaled Invariant Feature Transform* (SIFT), *Speed Up Robust Feature* (SURF), *Harris Detector* dan lain sebagainya. Salah satu metode *feature point detector* yang umum digunakan adalah *Harris Detector.*

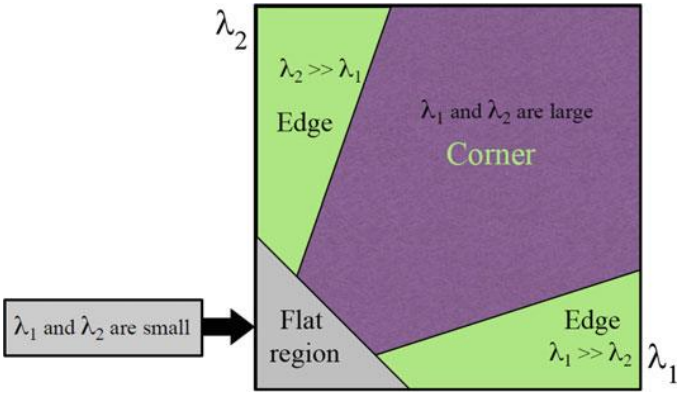
2.4.1. *Harris Detector*

Metode *Harris detectors* seringkali digunakan untuk mendeteksi keberadaan struktur sudut maupun tepi dalam gambar. Metode ini mampu mendeskripsikan sebuah gambar dalam bentuk tiga objek utama yakni *Edge* (tepi), *Corners* (sudut) dan *Flat* (datar). Metode ini memanfaatkan perbandingan variasi intensitas warna dari sebuah titik piksel. Secara umum, metode ini menggunakan konsep *single scale detectors.* Dalam penerapannya *harris detector* menggunakan *searching window* untuk mengidentifikasi jenis objek yang diolah. Proses klasifikasi jenis *feature point* menggunakan beberapa persamaan matriks yakni sebagai berikut,

$$M = w(x,y)\begin{pmatrix} \sum_{(x,y)} Ix^2 & \sum_{(x,y)} Ix\,Iy \\ \sum_{(x,y)} Ix\,Iy & \sum_{(x,y)} Iy^2 \end{pmatrix}) \tag{2.1}$$

Ix dan *Iy* merupakan hasil penurunan intensitas warna piksel pada sumbu x dan y. Sedangkan, *w(u,v)* merepresentasikan fungsi *weight window* sepanjang area (*u,v*)yang digunakan untuk mengidentifikasi *feature point*. Untuk menemukan *interest point*, *eigen*

value dari matriks M ditentukan untuk setiap piksel. Selanjutnya, *interest point* yang telah dideteksi akan diidentifikasi jenis objeknya berdasarkan *eigen value*. Objek sudut (*Corners*) direpresentasikan dalam bentuk selisih *eigen value* yang cukup besar. Diagram respon dari klasifikasi *interest point* berdasarkan *eigen value (λ)* yang korelasi dengan matriks M dapat dilihat pada Gambar 2.4.



Gambar 2.4 Klasifikasi *feature point* pada metode Harris *detectors* [1]

Untuk memperoleh diagram response *interest point* yang dideteksi, dilakukan proses kalkulasi parameter *cornerness measure* C(x,y) untuk setiap piksel [1]**.** Parameter tersebut merepresentasikan level dari *interest point* terhadap variasi intensitas warna disekitarnya. Berikut persamaan yang digunakan,

$$C(x,y) = \det(M) - K\left(trace(M)\right)^2,$$

$$\det(M) = \lambda 1 * \lambda 2\,, trace(M) = \lambda 1 + \lambda 2 \qquad (2.2)$$

Dengan $\lambda 1$ dan $\lambda 2$ merupakan nilai eigen dari variasi intensitas warna pada sumbu x serta sumbu y. K merupakan variable konstanta denga nilai antara 0,04 – 0,06. Metode ini telah banyak dikembangkan oleh para ahli untuk meningkatkan performa *feature point* Sedangkan, M merupakan variable matriks yang memuat variasi intensitas warna.

Salah satu hasil pengembangan metode tersebut adalah metode *Shi-Tomasi Corners Detector*.

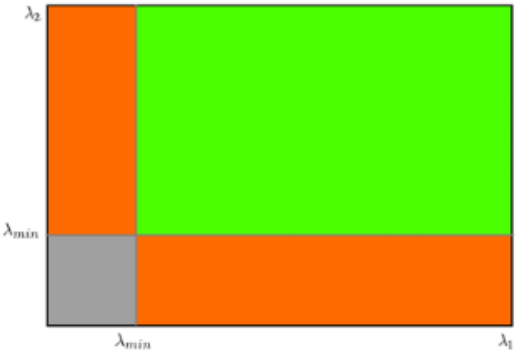
2.4.1.1. Metode Shi-Tomasi

Shi-Tomasi Corner Detector merupakan metode deteksi objek sudut yang dikembangkan oleh J.Shi dan C. Tomasi pada tahun 1994 [35]**.** Metode ini merupakan hasil modifikasi dari metode *Harris Detector***.** Namun, metode ini memiliki tingkat komputasi yang lebih sederhana. Objek sudut (*Corners*) direpresentasikan sebagai titik piksel yang memiliki selisih nilai *eigen* pada sumbu x dan y paling minimal. Sedangkan, untuk menentukan level sebuah objek sudut digunakan persamaan yakni sebagai berikut:

$$R = \lambda 1 * \lambda 2 - K(\lambda 1 + \lambda 2)^2 \quad (2.3)$$

$$R = Min(\lambda 1, \lambda 2) \quad (2.4)$$

R merupakan level respon suatu titik piksel terhadap struktur sudut. $\lambda 1$ dan $\lambda 2$ merupakan nilai eigen dari variasi intensitas warna pada sumbu x serta sumbu y. K merupakan variable konstanta denga nilai antara 0,04 – 0,06. Persamaan tersebut merupakan hasil modifikasi dari metode *Harris Detector*. Komputasi yang lebih sederhana membuat metode ini memiliki durasi waktu yang lebih singkat. Gambaran umum hasil deteksi *feature point* dengan metode Shi Tomasi dapat dilihat pada Gambar 2.5.



Gambar 2.5 Metode Shi Tomasi *Corners Detector* [35]

2.5. *Feature Point Matching*

Pada berbagai aplikasi *computer* vision, proses *feature point matching* digunakan untuk menemukan korespondensi terbaik (pasangan) dari sebuah *feature point* pada data gambar lain [1]. Data tersebut diperoleh dengan membandingkan data *feature point* dengan himpunan titik piksel pada data gambar yang dibandingkan. *Feature matching* atau *image matching* merupakan bagian dari sebagian besr aplikasi berbasis *Computer vision* seperti *image registration, camera calibration* dan *object recognition*, dalam tugas untuk mendeskripsikan hubungan antara dua gambar dengan objek yang sama [1]. Terdapat beberapa metode *feature matching* yang umum digunakan dalam proses analisa gambar. Metode tersebut antara lain *block matching* dan *optical flow matching*. Kedua metode tersebut merupakan metode yang umum digunakan dalam berbagai aplikasi analisa gambar.

2.5.1. *Optical Flow Matching*

Metode ini memanfaatkan aliran optic (*Optical flow*) dari sebuah *feature point* untuk mendekripsikan hubungan antara dua gambar. *Optical flow* merupakan pola dari gerak nyata sebuah objek dalam dua gambar yang saling berurutan, fenomena ini disebabkan oleh pergerakan dari objek tersebut maupun kamera [12]. Metode *optical flow matching* memanfaatkan beberapa asumsi kondisi *optical flow* untuk menentukan pasangan *feature point* dari dua gambar yang berurutan. Terdapat dua assumsi utama yang digunakan yakni sebagai berikut,

1. *Brightness constancy assumption*, dimana intensitas pixel suatu objek tidak berubah dalam dua gambar (frame) yang saling berurutan.
2. Piksel disekitar *feature point* dalam area tertentu memiliki gerak yang sama [13].

Berdasarkan dua assumsi tersebut, untuk mengidentifikasi pasangan *feature point* digunakan persamaan yakni sebagai berikut,

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.5}$$

Dalam proses identifikasi, digunakan algoritma tambahan untuk membantu menyelesaikan persamaan *optical flow* tersebut. Salah satu algoritma yang umum digunakan adalah metode Lucas Kanade. Hasil

akhir dari proses *Optical flow matching* berupa vector gerak dari dua *feature point* yang saling berkorelasi.

2.5.1.1. Lucas Kannade

Metode Lucas Kannade digunakan untuk memecahkan persamaan aliran optik dasar pada semua titik piksel dalam suatu lingkungan area dengan kriteria kuadrat yang kecil [14]. Metode ini tidak mampu memberikan informasi aliran optic dalam sebuah area gambar yang seragam. Metode Lucas kanade mengasumsikan bahwa lairan optic pada dasarnya bernilai konstan pada sebuah lingkungan local piksel yang sedang dipertimbangkan [14]. Persamaan aliran optik diasumsikan berlaku untuk semua titik piksel dalam sebuah area jendela (*window*) yang berpusat di sebuah titik P [15]. Aliran optik tersebut diwujudkan dalam bentuk vector kecepatan (Vx,Vy). Metode Lucas kanade mengassumsikan perubahan lokasi dari sebuah objek dalam gambar antar dua frame (gambar) yang saling berdekatan bernilai kecil dan mendekati dalam lingkungan di sekitar titik p [5]. Konsep tersebut dapat dituliskan sebagai berikut,

$$\begin{aligned} Ix(p1)Vx + Iy(p1)Vy &= -It(p1) \\ Ix(p2)Vx + Iy(p2)Vy &= -It(p2) \\ &\vdots \\ Ix(pn)Vx + Iy(pn)Vy &= -It(pn) \end{aligned} \tag{2.6}$$

Dengan P1, P2…,Pn merupakan titik piksel yang terdapat dalam area (*window*), Ix(Pn), Iy(Pn) dan It(Pn) merupakan nilai turunan parsial dari gambar I terhadap posisi x, y dan waktu t. Hal ini merepresentasikan hasil evaluasi terhadap point Pn dan waktu. Vx dan Vy merepresentasikan perubahan posisi titik piksel terhadap waktu. Persamaan tersebut dapat disederhanakan dalam bentuk matriks yakni sebagai berikut,

$$Av = b \tag{2.7}$$

$$A = \begin{bmatrix} Ix(p1) & Iy(p1) \\ Ix(p2) & Iy(p2) \\ \vdots & \vdots \\ Ix(pn) & Iy(pn) \end{bmatrix}, V = \begin{pmatrix} Vx \\ Vy \end{pmatrix}, b = \begin{bmatrix} -It(p1) \\ -It(p2) \\ \vdots \\ -It(pn) \end{bmatrix} \tag{2.8}$$

Melalui pendekatan persamaan *least square* dapat ditentukan solusi dari persamaan Lucas kanade diatas [5]. Dari persamaan diatas dapat disederhanakan menjadi,

$$V = (A^T A)^{-1} A^T b \tag{2.9}$$

Untuk menyelesaikan persamaan diatas, digunakan konsep pembebanan (*Weighted Version*) dari persamaan *least squares* [5]. Sehingga, diperoleh persamaan akhir untuk memperoleh vektor kecepatan (*Velocity Vector*) yakni sebagai berikut,

$$\sum w^2(x,y)[\nabla I\,(x,y,t)v + It(x,y,t)]^2 \tag{2.10}$$

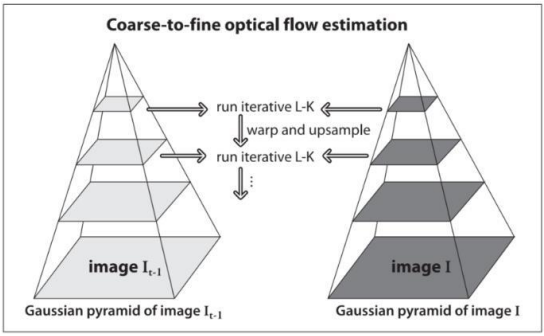
W(x,y) merupakan *window function* untuk memperoleh nilai perubahan posisi titik piksel terhadap satuan waktu. Penggunaan metode Lucas kanade seringkali diterapkan untuk menentukan pergerakan dari setiap target [15]. Konsep ini seringkali disebut dengan *Feature Point Tracking* (pelacakan *feature point*) berdasarkan metode Lucas Kanade.

2.5.1.2. *Image Pyramid*

Konsep *image pyramid* merupakan representasi dari kumpulan level resolusi gambar yang berbeda. Konsep ini digunakan untuk meningkatkan nilai akurat dan ketepatan pada proses *Optical Flow Matching*. Metode *Optical flow* dapat memberikan nilai akurat untuk pergerakan kecil berdasarkan hasil kalkulasi titik piksel terdekat. Namun, pergerakan yang cukup besar akan diabaikan [16]. Penggunaan metode *Image Pyramid* untuk proses *Optical Flow Matching* dapat meminimalisir keberadaan *Optical Flow Vector* yang tida berguna ketika melakukan proses perhitungan untuk perubahan posisi piksel bernilai cukup besar [16]. Untuk merancang sebuah *Image Pyramid*, sebuah gambar I dengan ukuran *nx* x *ny* sebagai gambar level 0. Gambar ini pada dasarnya adalah gambar dengan resolusi tertinggi (gambar mentah). Representasi piramida kemudian dibangun secara rekursif. Singkatnya, IL ditentukan berdasarkan IL − 1 di mana L adalah tingkat piramidal. Dalam proses perancangannya digunakan persamaan yakni sebagai berikut,

Dalam proses penentuan *feature point* yang saling terhubung, setiap *feature point* ditentukan titik yang korespon pada setiap level dalam *Image Pyramid* pada dua gambar yang saling berurutan. Gambaran umum dari proses *Optical Flow Matching* dengan *Image Pyramid* dapat dilihat pada Gambar 2.6.

$$I^L(x,y) = {}^1\!/_4\, I^{L-1}(2x,xy) + {}^1\!/_8\, I^{L-1}(2x-1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1) + {}^1\!/_{16}\, I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) \quad (2.11)$$



Gambar 2.6 Konsep *Image Pyramid* [16]

2.6. *Affine Transformation*

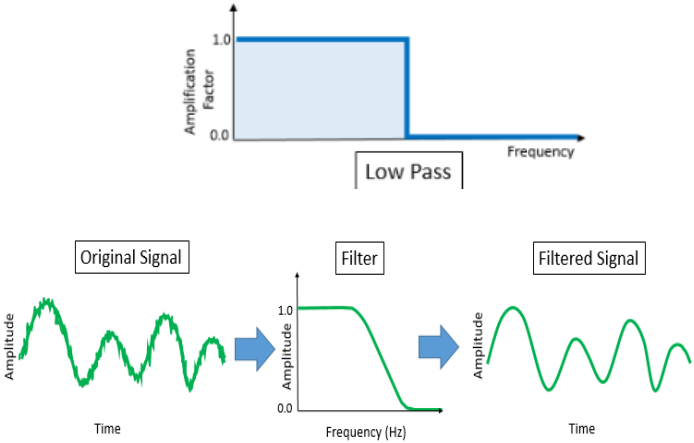
Metode *affine transformation* merupakan metode representasi gerak yang hampir mirip dengan *Simililarity Transformation* [17]. Namun, dalam jenis transformasi gerak ini turut terdapat representasi dari sudut yang menentukan arah penskalaan dan rasio parameter penskalaan [17]. Sehingga, dapat dikatakan bahwa transfomasi ini memiliki dua derajat kebebasan lebih banyak dari *Similarity transformation*. Transformasi ini merupakan kombinasi dari rotasi, penskalaan, translasi dan refleksi dari beberapa kasus [34]. Persamaan *Affine Transformation* dapat dituliskan yakni sebagai berikut,

$$T = \begin{bmatrix} \cos(\emptyset)\, s & -\sin(\emptyset)\, s & tx \\ \sin(\emptyset)\, s & \cos(\emptyset)\, s & ty \end{bmatrix} \quad (2.12)$$

Dengan tx dan ty merupakan representasi dari gerak translasi pada sumbu x dan sumbu y. Sedangkan, θ merupakan representasi dari gerak rotasi. Serta, S merupakan representasi dari gerak penskalaan.

2.7. *Simple Low Pass Filter*

Pada *simple low pass filter* digunakan konsep *Low Pass Filter* dengan pendekatan *Differential Equation* [18]. Secara umum, *Low Pass Filter* digunakan untuk melewatkan frekuensi rendah dan menghilangkan frekuensi tinggi dari sebuah sinyal data [19]. Sinyal data yang diolah tergantung dari aplikasi penggunaan dari filter. Konsep dari *Low Pass Filter* dapat dilihat pada Gambar 2.7.



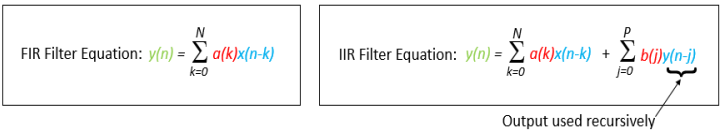
Gambar 2.7 Konsep *Simple low pass filter* [19]

Konsep *Simple Low Pass Filter* seringkali diterapkan dalam proses pengolahan data digital. Persamaan umum dari *Simple Low Pass Filter* dapat dituliskan yakni sebagai berikut:

$$y(n) = x(n) + x(n-1) \tag{2.13}$$

dengan *y(n)* merupakan sinyal amplitude keluaran dari filter. Sedangkan, x(n) merupakan amplitude dari sinyal masukkan yang akan diolah dengan n merupakan input sample data waktu. Filter ini meredam adanya perubahan nilai data secara tiba-tiba dalam sebuah sinyal data.

Secara tidak langsung, penggunaan filter ini mampu meredam frekuensi perubahan nilai data secara tiba-tiba dalam sebuah rentang aliran data tertentu. Perubahan nilai data secara tiba-tiba seringkali disebut dengan *Impulse Response*. Dalam filter data digital (*Digital Filters*) dapat dibedakan menjadi dua kelas berdasarkan *Impulse Response* yang diolah yakni *Finite Impulse Response* (FIR) dan *Infinite Response Response* (IIR) [19]. Istilah *Impulse Response* merujuk pada penampilan filter ketika berada pada domain waktu [19]. Konsep *Simple Low Pass Filter* turut dibedakan menjadi dua kelas berdasarkan *Impulse Response* dengan persamaan dapat dilihat pada Gambar 2.8.



Gambar 2.8 Klasifikasi *Simple low pass filter* berdasarkan *Impulse respons* [19]

2.8. *Video Stabilization Quality Assesment*

Salah satu cara membandingkan algoritma stabilisasi adalah dengan membandingkan kinerja aplikasi yang menggunakan video yang distabilkan [20]. Penilaian kualitas video yang telah distabilkan dapat merepresentasikan kinerja dari algoritma dan metode stabilisasi yang digunakan. Parameter ini seringkali dijadikan salah satu acuan untuk mengukur performa dari sebuah sistem stabilisasi video. Terdapat beberapa metode maupun parameter yang digunakan untuk mengukur kualitas sebuah video. Evaluasi subjektif merupakan teknik yang paling dapat diandalkan untuk menilai kualitas sebuah gambar maupun video. Hal ini dikarenakan keterlibatan manusia sebagai pengamat. Metode *Mean Opinion Score* (MOS) merupakan teknik evaluasi subjektif yang diperoleh dari sejumlah pengamat manusia [21]. Namun, teknik ini memiliki teknik komputasi yang rumit dan lambat untuk diterapkan dalam sebuah aplikasi. Untuk mengatasi hal tersebut, berbagai penelitian mengenai metode evaluasi objektif turut dikembangkan untuk mengukur kualitas gambar maupun video. Tujuan dari penelitian penelitian ini adalah merancang metrik kualitas yang dapat memprediksi kualitas gambar dan video yang dirasakan secara otomatis [21]. Metrik kualitas gambar obyektif dapat diklasifikasikan sesuai dengan

ketersediaan gambar asli (bebas distorsi) [22]. Gambar tersebut kemudian dibandingkan dengan gambar yang terdistorsi. Terdapat tiga jenis matrik kualitas gambar objektif yakni *Full Reference, No-reference* dan *Reduced-reference*. Matriks kualitas gambar objektif *Full Reference* memiliki gambar referensi yang komplit untuk diolah. Berikut beberapa parameter pada matriks kualitas video stabilisasi yakng digunakan,

2.8.1. *Image Quality Metric*

Image Quality (kualitas gambar) merupakan kesan subyektif yang ditemukan dalam pikiran pengamat yang berkaitan dengan tingkat keunggulan yang ditunjukkan oleh gambar [23]. Sedangkan, *Image Quality Matric* (matriks kualitas gambar) merupakan model yang dirancang untuk memprediksi kualitas gambar secara otomatis [24]. Biasanya kriteria otomatis untuk kualitas gambar didasarkan pada perbandingan piksel-demi-piksel sederhana antara gambar ground truth dan gambar referensi [33]. Hasil perbandingan tersebut pada umunya direpresentasikan dalam bentuk parameter seperti *Sum of Absolut Difference* (SAD), *Mean Squared Error* (MSE) dan *Peak to Signal Ratio* (PSNR). Namun, metode ini dianggap kurang effektif untuk mewakili persepsi manusia mengenai kualitas gambar. Dalam tiga hingga empat dekade terakhir, banyak upaya telah dilakukan untuk mengembangkan penilaian objektif untuk kualitas gambar dan video. Metode tersebut menggabungkan persepsi ukuran kualitas dengan mempertimbangkan karakteristik *Human Visual System* (HVS) [21]. *Human Visual System* (HVS) sendiri terdiri dari dua bagian yakni mata dan otak. Perlu digaris bawahi, bahwa manusia tidak merasakan gambar sebagai sinyal dalam ruang dimensi tinggi. Namun, lebih tertarik dalam berbagai atribut gambar-gambar itu, seperti kecerahan, kontras, bentuk dan tekstur objek, orientasi, kehalusan dan lain sebagainya [21]. Salah satu parameter yang umum digunakan untuk merepresentasikan kualitas gambar berdasarkan persepsi HVS adalah *Structural Similarity Index Matric* (SSIM).

2.8.1.1. *Mean Squared Error* (MSE)

Parameter MSE merupakan matriks yang memuat kesalahan kuadrat kumulatif antara gambar asli dan gambar terdistorsi [24]. Nilai kesalahan (error) diperoleh dari hasil perbandingan intensitas piksel antara gambar asli dan gambar yang terdistorsi. Berikut persamaan matematika dari MSE,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \quad (2.14)$$

Karena kesederhanaannya, MSE (*mean square error*) menjadi salah satu parameter pada matriks kinerja kuantitatif yang umum digunakan dibidang pemrosesan sinyal selama bertahun-tahun [25].

2.8.1.2. *Sum of Absolut Difference* (SAD)

Ukuran distorsi blok paling umum (BDM) adalah *Sum of Absolut Difference* (SAD) yang memberikan hasil hampir serupa dengan MSE tetapi dengan lebih sedikit perhitungan [25]. Berikut persamaan matematika untuk mendapatkan parameter tersebut,

$$SAD_{\ x,y}(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|F_t - F_{t-1}(x+i+u, y+j+v)| \quad (2.15)$$

Dengan N merupaka ukuran gambar, (*u,v)* merupakan perpindahan posisi dari posisi (*x,y*).

2.8.1.3. *Structural Similarity Index Matric* (SSIM)

Parameter ini mendefinisikan informasi struktural dalam gambar sebagai atribut yang mewakili struktur objek dalam adegan tertentu. Hal ini terlepas dari luminance dan kontras rata-rata. Indeks SSIM didasarkan pada kombinasi pencahayaan, kontras, dan perbandingan struktur [23]. Seperti yang diketahui, Luminansi (*Luminance*) permukaan suatu objek yang sedang diamati adalah produk dari iluminasi dan pantulan cahaya. Namun, struktur objek dalam gambar bersifat independen terhadap iluminasi [26]. Berikut persamaan matematika dari indeks SSIM yang dikembangkan oleh Amin [25],

$$S(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x{}^2 + \mu_y{}^2 + C_1}\right)\left(\frac{2\sigma_X\sigma_Y + c_2}{\sigma_X{}^2 + \sigma_y{}^2 + C_2}\right)\left(\frac{\sigma_{xy} + C_3}{\sigma_x\,\sigma_y + C_3}\right) \quad (2.16)$$

Dengan μx dan μy adalah sampel local dari x dan y, σx dan σy adalah sampel local standar deviasi x dan y, dan σxY adalah sampel korelasi silang x dan y setelah menghilangkan rata-ratanya [25]. Sedangkan, C1,C2 dan C3 merupakan konstanta. Persamaan tersebut memperhitungkan struktur suatu objek dalam berbagai jenis gangguan. Melalui parameter SSIM diperoleh sebuah nilai kesamaan (*similarity*)

struktur antara kedua gambar terlepas dari pengaruh efek luminansi dan lain sebagainya.

2.9. Pemograman C++ dan *Object Oriented Programming* (OOP)

Bahasa C merupakan pengembangan dari bahasa B yang ditulis oleh Ken Thompson pada tahin 1970. Bahasa C untuk pertama kali ditulis oleh Brian W. Kernighan dan Denies M. Richtie pada tahun 1972 [27]. Pada awalnya bahasa C dioperasikan diatas sistem operasi UNIX [27]. Pada tahun 1980, seorang ahli bernama Bjarne Stroustrup mengembangkan beberapa hal dari bahasa C yang dinamakan "*C with Classes*". Pada tahun 1983, konsep tersebut berganti nama menjadi C++ [27]. Pada pemograman C++ terdapat sebuah konsep yakni *Object Oriented Programming* (OOP). Konsep OOP bertujuan untuk membantu pengguna dalam merancang dan mengelola program yang besar dan kompleks [27]. Ide utama dari konsep OOP adalah program yang disusun sebagai kumpulan objek yang saling berinterasi dan setiap objek memiliki ruang dan fungsi data [28]. Analisis dan desain program berorientasi objek dipusatkan di sekitar objek data, fungsi yang terkait dengan setiap objek, dan hubungan antara objek [28]. Pada konsep pemograman OOP terdapat dua konsep penting yakni *class* (kelas) dan objek. *Class* merupakan unit utama pemrograman dalam bahasa berorientasi objek [28]. *Class* berisi elemen data yang disebut dengan *member* dan kumpulan tugas maupun fungsi yang disebut dengan *methods.*

2.10. *Framework* Qt

Aplikasi Qt merupakan *cross-platform framework* yang seringkali digunakan sebagai *toolkit* grafis dan perancangan aplikasi [29]. Sistem Qt sendiri mendukung untuk diterapkan pada berbagai *platform* dan *Operating System* (OS). Qt sendiri memiliki beberapa modul antara lain QtCore, QtGui dan QtWidgets, Qt Network, QtWebkit, QtSQ, QtXML dan QtXmlPatterns. Salah satu penerapan dari penggunaan *framework* tersebut digunakan untuk merancang berbagai aplikasi. Dalam proses installasinya, aplikasi ini turut dilengkapi dengan *packages* Qt Creator, QMake Documentation, Qt Documentation dan Qt 4.8.1 (Dekstop). Qt Creator merupakan salah satu IDE (*Integrated*

Development Environment) untuk C++, namun dapat digunakan untuk pengkodean pada aplikasi Qt. Ikon aplikasi dapat dilihat pada Gambar 2.9.

Gambar 2.9 *Framework* Qt [29]

2.11. *Library* OpenCV

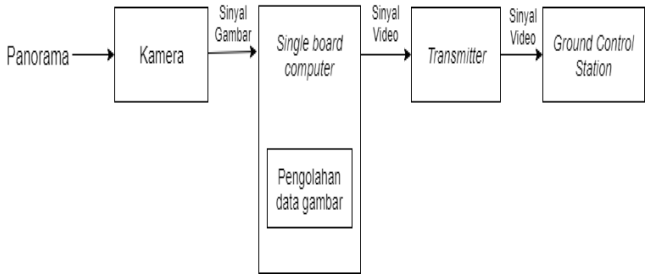
Merupakan *library* untuk *digital image processing* yang dirancang oleh inte dan dikembangkan oleh Willow Garage. Tersedia untuk pemograman C++, C dan Phyton. *Library* ini bersifat *Open source* dan dalam penggunananya tidak dikenai biaya. *Library* ini seringkali digunakan untuk merancang sebuah aplikasi yang memerlukan pengolahan data gambar. Versi terbaru dari *library* adalah versi 2.2. Tampilan logo *library* OpenCV dapat dilihat pada Gambar 2.10.

Gambar 2.10 *Library* OpenCV [38]

Halaman ini sengaja dikosongkan

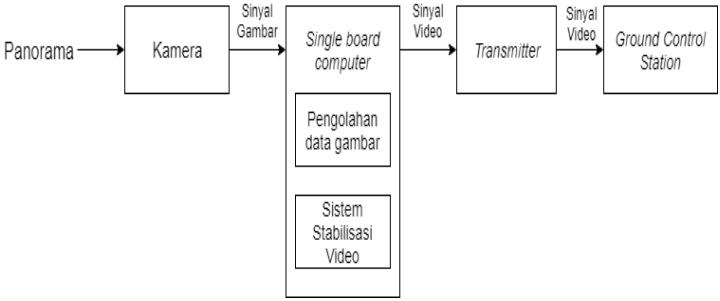
BAB III
PERANCANGAN SISTEM STABILISASI VIDEO DAN MODUL UJI PERFORMA

Pada bab ini memuat keseluruhan tahap perancangan sistem stabilisasi video dan modul uji untuk pengujian performa sistem stabilisasi video. Sistem stabilisasi video memuat keseluruhan algoritma metode pengolahan data gambar untuk meredam efek gerak getar pada video. Sistem stabilisasi video diimplementasikan dalam sistem pengolahan gambar pada UAV. Diagram alir dari sistem pengolahan data gambar dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram blok sistem pengolahan data gambar hasil tangkapan kamera

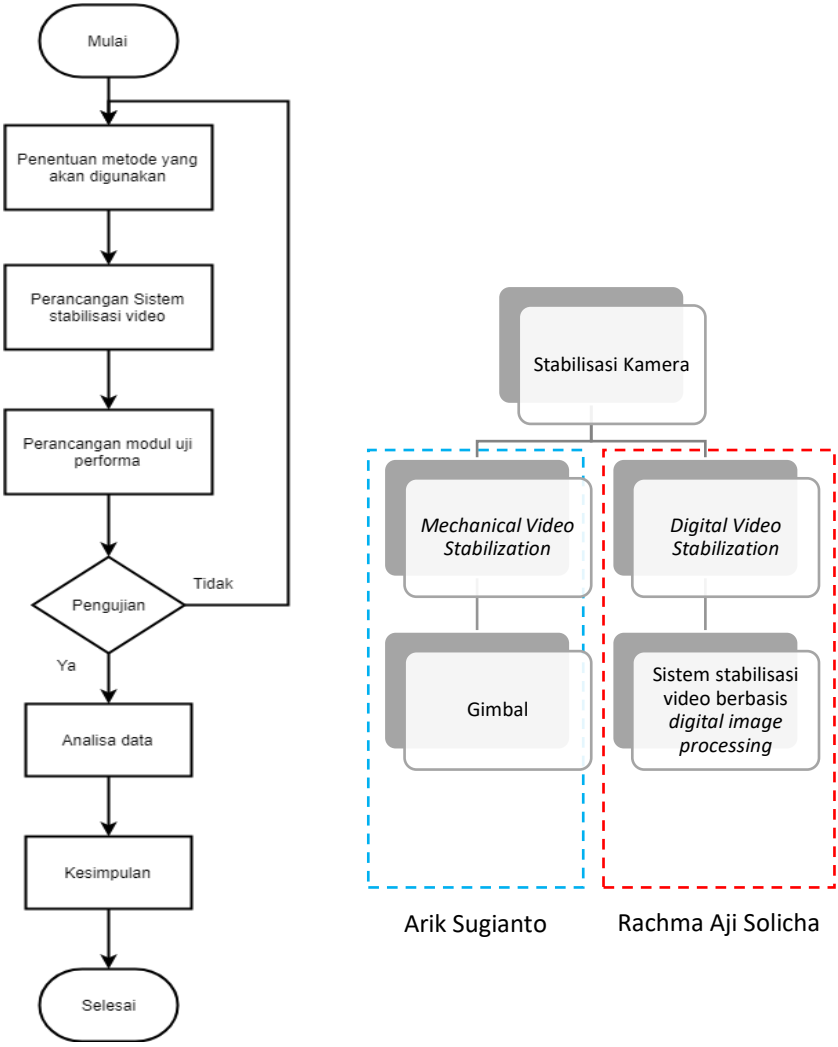
Pada Gambar 3.1 dapat dilihat aliran data pada sistem pengolahan data gambar hasil tangkapan kamera. Perangkat *single board computer* bertugas untuk mengakses kamera dan mengolah data gambar hasil tangkapan kamera tersebut. Hasil akhir proses pengolahan data gambar melalui perangkat *transmitter* dikirimkan ke perangkat *Ground Control Station* (GCS)**.** Sistem stabilisasi video yang dikembangkan betugas untuk mendeteksi dan meredam gangguan pada tampilan video yang dihasilkan. Sistem tersebut diimplementasikan pada perangkat *single board computer* untuk mengolah data gambar hasil tangkapan kamera. Gambaran umum dari implementasi sistem stabilisas video dapat dilihat pada Gambar 3.2.



Gambar 3.2 Implementasi sistem stabilisasi video pada sistem pengolahan data ganbar UAV

Sistem stabilisasi video menerapkan metode *Digital Video Stabilization*. Metode ini memanfaatkan keberadaan pelacakan *feature point* pada gambar untuk mendeteksi adanya gangguan pada tampilan video. Selain itu, sistem stabilisasi video dirancang untuk mampu mengolah data hasil tangkapan kamera secara *real-time.* Hal ini disesuaikan dengan kebutuhan fitur *video streaming* pada UAV. Terdapat beberapa tahapan dalam proses perancangan sistem stabilisasi video seperti pada Gambar 3.3 (a). Perancangan sistem stabilisasi video merupakan bagian dari proyek fitur stabilisasi kamera untuk produk UAV kedepannya. Pembagian kerja dalam proyek ini dapat dilihat pada Gambar 3.3 (b).

Pada Gambar 3.3 (a) dapat dilihat proses pengerjaan proyek ini diawali dengan penentuan metode pengolahan data video. Kumpulan metode pengolahan video kemudian dirangkai dalam sebuah sistem pengolahan data gambar. Proses perancangan sistem stabilisasi video menggunakan *framework* Qt.5 dan *library* OpenCV. Hasil akhir dari proses stabilisasi video ditampilkan dalam sebuah *Graphical User Interface* (GUI). Proses pengujian performa sistem stabilisasi video dilakukan dengan bantuan modul uji. Modul uji tersebut memuat metode pengukuran parameter pengujian sistem stabilisasi video. Hasil pengujian tersebut digunakan sebagai bahan evaluasi untuk meningkatkan performa sistem stabilisasi video yang tengah dikembangkan.



(a) (b)

Gambar 3.3 Proyek stabilisasi kamera, (a) Diagram alir tahap pengerjaan proyek (b) Pembagian ranah kerja

Berikut spesifikasi dari sistem stabilisasi video dan modul performa yang dikembangkan,

3.1. *Design Requirement*

Gangguan pada proses pengambilan gambar oleh kamera UAV seringkali mengakibatkan efek getar pada tampilan video. Gangguan tersebut dapat berasal dari getaran mesin UAV, gesekan udara hingga pergerakan UAV itu sendiri. Di lain sisi penggunaan fitur *video streaming* pada UAV memungkinkan pengguna untuk mengintai setiap pergerakan target secara *real-time*. Pada penerapannya, durasi pengolahan gambar berpengaruh besar terhadap kualitas video yang dihasilkan. Oleh karena itu, diperlu9kan sebuah konsep sistem stabilisasi video yang mampu mengatasi berbagai gangguan tersebut serta memiliki durasi pengolahan yang singkat. Berdasarkan uraian tersebut, terdapat beberapa target performa dan spesifikasi dari sistem stabilisasi video yang perlu dicapai. Target performa tersebut antara lain:

- Durasi pemrosesan data gambar berkisar 0.033 sekon per gambar (frame video) atau dapat dikatakan kecepatan pengolahan data frame video berkisar 30 FPS (*frame per second*).
- Dapat meningkatkan kualitas video dengan meredam gangguan berupa gerak getar maupun perubahan sudut pengambilan gambar.

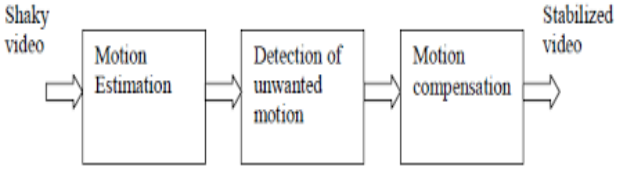
Pada proses pengujian sistem stabilisasi video, terdapat beberapa parameter performa yang perlu diukur. Hasil dari proses pengujian digunakan sebagai referensi dalam meningkatkan performa sistem stabilisasi video. Berdasarkan uraian diatas, maka dalam perancangan modul uji terdapat beberapa target spesifikasi sistem yang perlu dicapai yakni sebagai berikut:

- Dapat mengukur parameter performa sistem stabilisasi video.
- Dapat menyimpan hasil pengukuran parameter untuk analisa performa sistem stabilisasi video.
- Dapat menampilkan beberapa hasil pengukuran parameter pengujian performa sistem stabilisasi video.

3.2. Desain Sistem Stabilisasi Video

Sistem stabilisasi video yang dikembangkan menerapkan metode *digital video stabilization*. Setiap data gambar dianalisa dan diolah untuk memperoleh tampilan gambar yang lebih baik. Secara

umum*n*, terdapat beberapa tahap pengolahan data gambar yang diterapkan dalam suatu proses stabilisasi video. Diagram blok tahap pada proses stabilisasi video dapat dilihat pada Gambar 3.4.

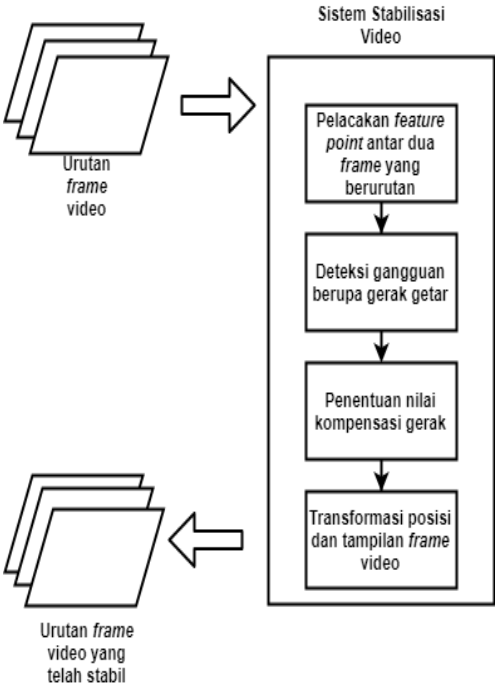


Gambar 3.4 Diagram blok tahap pengolahan data gambar pada metode *digital video stabilization*

Pada Gambar 3.4 data video yang tidak stabil (*shaky video*) diolah pada tahap estimasi gerak untuk memperoleh data pergerakan kamera. Data pergerakan tersebut disaring pada tahap deteksi gerak yang tidak diinginkan. Efek gerak yang tidak diinginkan selanjutnya diredam pada tahap kompensasi gerak.

Pada teknik *digital video stabilization*, terdapat penambahan tahap *image pre-processing* dan *image matching*. Penambahan tahap *image pre-processing* bertujuan untuk memberikan kondisi gambar yang ideal untuk proses pengolahan berikutnya. Sedangkan, penambahan tahap *image matching* bertujuan untuk mendeskripsikan hubungan antara dua *frame* video yang saling berurutan. Sehingga, keseluruhan konsep sistem stabilisasi video dapat dilihat pada Gambar 3.5.

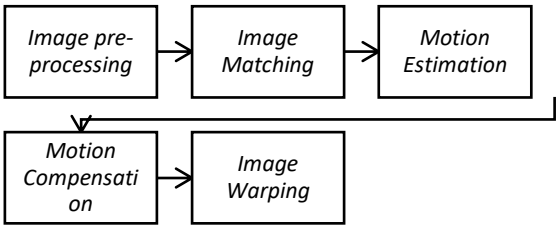
Sistem stabilisasi video turut dilengkapi dengan *Graphical User Interface* (GUI). Penambahan GUI bertujuan untuk mempermudah pengguna dalam mengamati hasil proses stabilisasi video, serta memberikan akses ke pengguna dalam menentukan data video yang diolah. Dalam proses perancangan sistem stabilisasi video menggunakan framework Qt.5. Sedangkan, untuk proses pengolahan gambar memanfaatkan fungsi yang terdapat dalam library OpenCV. Proses perancangan sistem stabilisasi video terdiri dari dua tahap yakni perancangan tahap pengolahan data gambar dan Graphical User Interface (GUI). Berikut rincian proses perancangan program sistem stabilisasi video,



Gambar 3.5 Konsep sistem stabilisais video

3.2.1. Tahap Pengolahan Data *Frame* Video

Data file video diolah menggunakan bantuan fungsi *image processing* yang terdapat dalam *library* OpenCV. Proses pengolahaN data file video dapat dilihat pada Gambar 3.6.



Gambar 3.6 Tahap pengolahan data *frame* video

Pada Gambar 3.6 dapat dilihat bahwa proses pengolahan data gambar terdiri dari beberapa tahap. Hasil akhir pengolahan data gambar berupa data gambar dikirimkan melalui komunikasi *Signal* dan *Slots* ke fitur yang terdapat dalam GUI. Berikut rincian program proses pengolahan data file video,

3.2.1.1. *Image pre-Processing*

Tahap ini terdiri dari dua proses utama yakni konversi data *frame* video dari format warna *Red*, *Green* dan *Blue* (RGB) menjadi *Grayscale* (abu-abu) dan deteksi *feature point.* Dalam proses konversi format warna data gambar digunakan bantuan *library* OpenCV berupa fungsi "cvtColor()". Pada proses ini, kedua data *current frame* dan *previous frame* dikonversi menjadi format *grayscale* sebelum memasuki tahap pengolahan berikutnya. Proses konversi format warna data *frame* video dilakukan berdasarkan kebutuhan dari sebagian besar metode yang diterapkan dalam fungsi stabilisais video. Instruksi yang digunakan dapat dilihat pada Gambar 3.7.

```
Mat prev_grey;
cvtColor (prev, prev_grey,COLOR_BGR2GRAY);
```

Gambar 3.7 Instruksi program untuk konversi warna *frame* video

Data *frame* video kemudian memasuki tahap deteksi *feature point*. Pada fungsi stabilisasi video yang dikembangkan, *feature point* yang digunakan berupa struktur sudut (*Corners*) dari objek dalam *frame* video. Struktur sudut (*corners*) dianggap sebagai *feature point* yang paling intuitif [30]. Perubahan kecil dari sebuah objek dibawah intensitas cahaya rendah sekalipun dapat dideteksi dengan mengobservasi kondisi dari struktur sudut (*corners*) objek dalam gambar [30]. Dalam proses deteksi *feature point* digunakan metode *shi-thomasi corners detector*. Metode ini memiliki algoritma komputasi yang sederhana, jika dibandingkan dengan metode *Harris corners detectors.* Serta, mampu memberikan hasil deteksi dengan kualitas yang cukup bagus. Penggunaan metode ini memberikan durasi waktu pemrosesan yang cukup singkat. Hal ini sesuai dengan kebutuhan sistem stabilisasi video yakni pengolahan data *real-time*.

Dalam penerapannya, proses deteksi *feature point* memanfaatkan fungsi "goodFeaturestoTrack()" yang terdapat dalam *library* OpenCV. Proses deteksi *feature point* hanya diterapkan pada data *previous frame*. Hal ini dilakukan untuk memperoleh data kondisi awal (*initial condition*) dari objek yang terdapat dalam *frame* video. Instruksi yang digunakan dalam proses deteksi *feature point* dapat dilihat pada Gambar 3.8.

```
vector <Point2f> prev_corner,
                 cur_corner;
goodFeaturesToTrack (prev_grey,
                     prev_corner,100,0.01,
                     30,noArray(),5,
                     false,0.04);//detct
                     corner
```

Gambar 3.8. Instruksi program untuk deteksi *feature point*

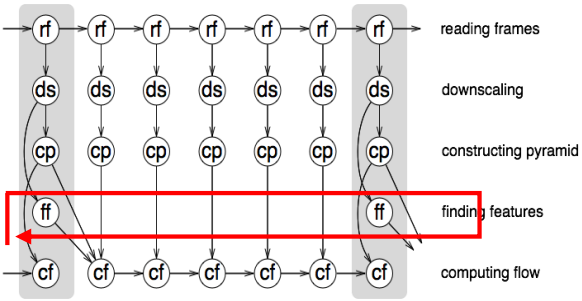
Pada Gambar 3.8 dapat dilihat bahwa untuk melakukan proses deteksi *feature point* terdapat beberapa parameter yang harus dipenuhi. Parameter tersebut antara lain yaitu jumlah maksimal *feature point* yang dideteksi, batas minimal kualitas dari *feature point*, ukuran *window* untuk daerah deteksi, dan jarak antara *feature point* yang dideteksi. Berikut spesifikasi yang digunakan,

- Jumlah maksimal *feature point* yang dideteksi: 100
- Batas minimal kualitas *feature point* : 0.01
- Jarak antara *feature point* : 30
- Ukuran daerah deteksi (*window size*) : 5

Parameter tersebut disesuaiakan dengan kebutuhan fungsi stabilisasi video.

Hasil akhir dari proses deteksi *feature point* berupa kumpulan data titik piksel yang ditampung dalam sebuah vector. Pada proses deteksi *feature point* turut diterapkan konsep optimalisasi yang dikembangkan oleh Serafing Chekalkin. Untuk meminimalisir durasi waktu pemrosesan, proses deteksi *feature point* dilakukan setiap

pembacaan enam *frame* sekali. Proses ini dapat digambarkan seperti pada gambar 3.9.



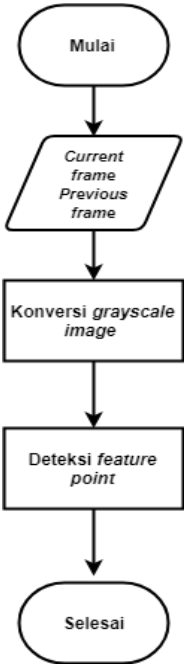
Gambar 3.9 Optimalisasi deteksi *feature point*

Menurut Serafig Chekalkin, jika pada *frame* sebelumnya *feature point* dianggap cukup baik (kualitas sudut tinggi) maka dapat dikatakan setelah melalui proses pemetaan kualitas *feature point* tetap baik [31]. Diagram alir dari keseluruhan proses pengolahan dapat dilihat pada gambar 3.10. Penggunaan metode ini bertujuan untuk meminimalisir durasi waktu pemrosesan. Instruksi lengkap dari program *image pre-processing* dapat dilihat pada Lampiran A-2.

3.2.1.2. *Image Matching*

Tahap ini memuat proses pemetaan *feature point* antara dua *frame* video yang saling berurutan. Proses pemetaan dilakukan untuk menentukan posisi *feature point* pada *current frame*. Proses pemetaan ini menggunakan metode pelacakan *feature point*. Sistem secara otomatis menentukan kumpulan titik piksel pada *curret frame* yang koresponden dengan *feature point*. Proses ini bertujuan untuk melihat hubungan antara *previous frame* dan *current frame* video. Hubungan tersebut berupa perubahan posisi objek yang ditandai dengan perubahan posisi *feature point*. Metode pelacakan *feature point* yang digunakan adalah *Optical flow matching* dengan metode Lucas kanade untuk komputasinya.

Metode ini memperhitungkan aliran optik (*optical flow)* antara dua titik piksel. Pasangan titik piksel dianggap memiliki aliran optik jika perbedaan nilai intensitas warna mendekati nol (konstan). Pada proses ini turut digunakan pendekatan *Image Pyramids* untuk meningkatkan nilai ketepatan (*robust*) dari proses pelacakan *feature point*.



Gambar 3.10 Diagram alir proses pengolahan data tahap *image pre-processing*

Metode pendekatan *Image Pyramids* memungkinkan sistem untuk melacak keberadaan *feature point* pada level resolusi gambar yang berbeda. Proses ini bertujuan untuk meredam adanya kesalahan ketika dihadapkan dengan perubahan posisi *feature point* yang cukup besar. Keberadaan aliran optik menunjukkan arah serta besar perubahan posisi *feature point* diantara dua *frame* vide yang berurutan. Hasil akhir dari proses ini berupa vector gerak (*motion vector*). Vektor gerak (*motion vector*) digunakan untuk proses estimasi gerak pada tahap *motion estimation*.

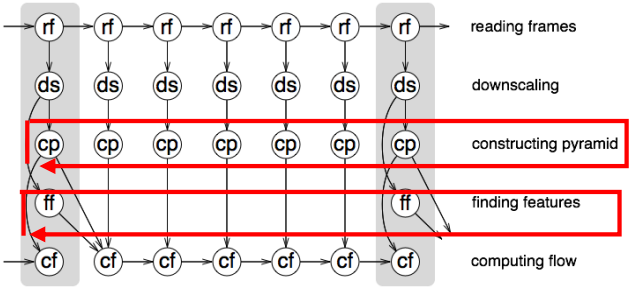
Proses ini memanfaatkan fungsi "calcOpticalFlowPyrlK ()" yang terdapat dalam *library* OpenCV. Fungsi tersebut digunakan untuk menghitung *Optical flow* dari sekumpulan *feature point* dengan metode Lucas Kanade dan menggunakan pendekatan *Image Pyramid.* Fungsi tersebut memiliki beberapa parameter yakni ukuran area deteksi

(*window size*) dan level *Image Pyramids*. Pada sistem stabilisasi video, digunakan parameter dengan spesifikasi yakni sebagai berikut:

- Ukuran area deteksi (*window size*) : 21 x 21 piksel
- Level *Image Pyramids* : 3

Pemilihan spesifikasi tersebut berdasarkan rekomendasi dari dokumentasi OpenCV dan hasil pengamatan pada pemberian spesifikasi yang berbeda.

Proses perhitungan aliran optik diawali dengan perancangan *Image Pyramids* pada *previous frame* dan *current frame* video. Hasil perancangan tersebut di inisialisasi sebagai *previous pyramids* dan *current pyramids*. Dalam proses perancangan *image pyramids* digunakan fungsi "buildOpticalflow()" yang terdapat dalam *library* OpenCV. Fungsi tersebut menggunakan parameter ukuran area deteksi (*window size*) dan level *image pyramids* dengan spesifikasi nilai yang sama dengan fungsi sebelumnya. Konsep penggunaan kedua instruksi tersebut dapat dilihat pada Gambar 3.11.



Gambar 3.11 Optimalisasi tahap *Image matching*

Proses ini dilakukan ketika sistem mencapai pembacaan *frame* video urutan kedua. Ketika pembacaan *frame* video berikutnya, sistem hanya melakukan perancangan *image pyramids* pada data *current frame* video. Sedangkan, untuk data *previous pyramids* menggunakan data *current pyramids* pada pembacaan *frame* video sebelumnya. Hasil akhir dari proses ini kemudian digunakan untuk proses perhitungan aliran optik menggunakan fungsi "calcOpticalPyrLk()". Instruksi yang digunakan pada tahap ini dapat dilihat pada Gambar 3.12.

Instruksi lengkap dapat dilihat pada Lampiran A-2. Keseluruhan proses pengolahan data gambar pada tahap ini dilihat pada Gambar 3.13.

```
vector<Mat>PrevPyramid,CurPyramid;
buildOpticalFlowPyramid (prev_grey,
            PrevPyramid,Size(21,21),3,true)
            ;
buildOpticalFlowPyramid (cur_grey,
            CurPyramid,Size(21,21),3,true);
calcOpticalFlowPyrLK ( PrevPyramid,
            CurPyramid,prev_corner,cur_corn
            er,status,err,Size(21,21),3);//
            Optical Flow
```

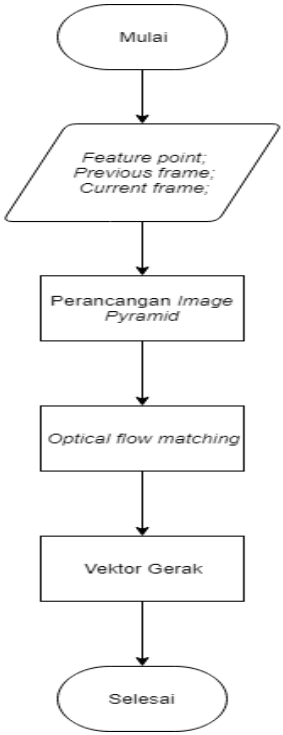
Gambar 3.12 Instruksi yang digunakan pada tahap *Image matching*

3.2.1.3. *Motion Estimation*

Pada tahap ini, dilakukan proses estimasi nilai pergerakan objek berdasarkan perubahan posisi *feature point* antara dua frame yang saling berurutan. Perubahan posisi dari *feature point* diidentifikasi sebagai bentuk transformasi gerak 2 dimensi berupa *Affine Transform* (subbab 2.6). Parameter pergerakan objek yang diukur terdiri dari gerak *translasi* (pergeseran) pada sumbu x, gerak *translasi* (pergeseran) pada sumbu y dan gerak rotasi. Dalam menentukan ketiga nilai parameter pergerakan objek digunakan fungsi "estimateRigidTrasform ()" pada *library* OpenCV. Jenis *affine transform* yang digunakan adalah *half affine transform*. Hasil pengukuran gerak objek menggunakan parameter transformasi tersebut berupa matriks 2 x 3 yang memuat ketiga nilai parameter gerak diatas. Matriks transformasi dapat dilihat pada persamaan yakni sebagai berkut,

$$M = \begin{bmatrix} \cos(\emptyset) & -\sin(\emptyset) & dx \\ \sin(\emptyset) & \cos(\emptyset) & dy \end{bmatrix} \quad (3.1)$$

Instruksi yang digunakan dalam tahap ini dapat dilihat pada Gambar 3.14.

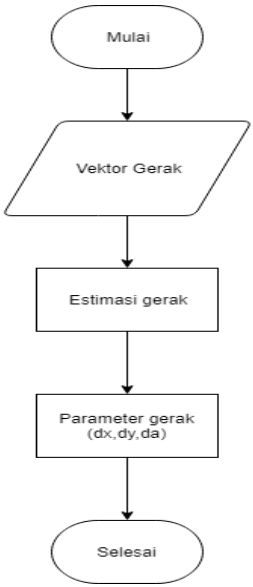


Gambar 3.13 Diagram alir tahap *Image matching*

```
// Estimasi gerak
Mat T= estimateRigidTransform
       (prev_corner2,cur_corner2,
       false);
// Parameter gerak
         dx = T.at<double>(0,2);
         dy = T.at<double>(1,2);
         da = atan2(T.at<double>(1,0),
             T.at<double>(0,0));
```

Gambar 3.14 Instruksi untuk estimasi gerak

Hasil akhir dari proses pengolahan data *frame* video pada tahap ini berupa nilai pergeseran objek di sumbu x (dx) dan disumbu y (dy), serta sudut rotasi (da). Perubahan nilai secara tiba-tiba dalam rentang periode waktu tertentu dapat diidentifikasi sebagai gangguan berupa gerak getar (*jitter*) maupun perubahan sudut pengambilan gambar. Keseluruhan proses pengolahan data *frame* video dapat dilihat pada Gambar 3.15.



Gambar 3.15 Diagram alir pengolahan data pada tahap *Motion estimation*

Keseluruhan instruksi program untuk tahap estimasi gerak dapat dilihat pada Lampiran A-2.

3.2.1.4. *Motion Compensation*

Pada tahap ini memuat keseluruhan proses pengolahan data hasil estimasi gerak objek dalam tampilan *frame* video. Tahap ini bertujuan untuk memperoleh parameter transformasi gerak yang mampu meredam gangguan dalam tampilan *frame* video. Parameter tersebut terdiri dari gerak *translasi* pada sumbu x (dx), gerak *translasi* pada

sumbu y (dy) dan gerak rotasi pada sudut tertentu (da). Ketiga parameter hasil estimasi gerak kemudian diolah untuk memperoleh nilai parameter transformasi gerak yang baru. Metode pengolahan data yang diterapkan merupakan adaptasi dari metode yang dikembangkan oleh Jinhai Chai dan Rodnet walker [32] dalam paper "*Robust Video Stabilization Algorithm using Feature Point Selection and Delta Optical Flow"*. Keseluruhan proses pengolahan data estimasi gerak dapat dilihat pada Gambar 3.16.

Proses pengolahan data diawali dengan memperoleh nilai akumulasi dari setiap parameter gerak objek. Pada metode yang dikembangkan oleh Jinhai Chai dan Rodnet Walker, nilai akumulasi parameter pergerakan objek diperoleh dengan menjumlahkan keseluruhan data sebelum posisi *frame* video saat ini. Berikut persamaan yang digunakan untuk menentukan nilai akumulasi gerak dalam metode tersebut,

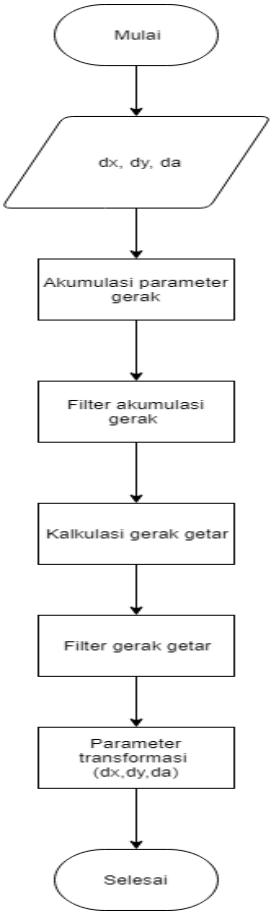
$$acl_{[t]} = \sum_{int\ i=0}^{t-1} x[t-i] \tag{3.2}$$

Penggunaan metode tersebut hanya mampu memberikan hasil yang effektif pada batas data tertentu. Sedangkan, sistem stabilisasi video nantinya digunakan untuk mengolah data *real-time* yang tidak terbatas oleh waktu. Oleh karena itu, dilakukan beberapa penyesuaian dalam penerapan metode tersebut. Pada proses penjumlahan data estimasi gerak dibatasi hingga rentang data ke 11 dari posisi data *frame* video saat ini. Persamaan 3 digunakan untuk menentukan nilai akumulasi gerak.

$$acl_{[t]} = \sum_{int\ i=0}^{10} x[t-i] \tag{3.3}$$

Instruksi program yang digunakan untuk memperoleh data akumulasi gerak pada masing-masing parameter gerak dapat dilihat pada Gambar 3.17. Hasil akumulasi dari masing-masing parameter gerak akan melalui proses filter data. Proses ini bertujuan untuk mendeteksi sekaligus meredam keberadaan perubahan nilai gerak secara tiba-tiba (*impulse*). Gangguan pada tampilan *frame* video berupa gerak getar (*jitter*) dapat diindikasi dengan keberadaan *impulse* secara terus-menerus dalam rentang periode waktu tertentu.

Metode filter data yang dikembangkan oleh Jinhai Chai dan Rodnet walker [32] menggunakan *Simple Low Pass Filter* dengan pendekatan *Infinite Impulse Response* (IIR).



Gambar 3.16 Diagram alir pengolahan data tahap *Motion compensation*

```
// Variable akumulasi gerak
double sum_x, sum_y, sum_a;
// Akumulasi gerak
 for(int i=-11; i < -1; i++){
     int a= k+i;
       if (a < 0){
             sum_x +=0;
             sum_y +=0;
             sum_a +=0; }
       else{
              sum_x +=_x[a];
              sum_y +=_y[a];
              sum_a +=_a[a];}
            }
```

Gambar 3.17 Instruksi program akumulasi gerak

Pada fungsi stabilisasi video turut diterapkan metode filter akumulasi data yang sama. Pada penerapan kedepannya, fungsi tersebut dirancang untuk mampu mengatasi gangguan gerak getar tanpa terbatas waktu. Hal ini sesuai dengan karakteristik *filter* yang dirancang untuk mengatasi respon *impulse* tanpa terbatas waktu (*Infinite Impulse Respon*). Berikut persamaan filter data yang digunakan,

$$SMA[t] = 0.95 * SMA[t-1] + 0.05 * acl\,[t] \quad (3.4)$$

Instruksi program yang digunakan untuk memperoleh hasil filter akumulasi parameter gerak objek dapat dilihat pada Gambar 3.18.

Proses pengolahan data dilanjutkan dengan kalkulasi nilai gerak getar (*jitter)*. Nilai gerak getar dapat di indentifikasi sebagai perbedaan antara nilai estimasi dan nilai gerak sebenarnya (*intentional motion*). Pada metode yang dikembangkan oleh Jinhai Chai dan Rodnet walker [32], Nilai gerak sebenarnya diasumsikan sebagai selisih antara nilai akumulasi gerak sebelum dan sesudah proses filter data. Berikut persamaan untuk memperoleh nilai kalkulasi gerak getar (*jitter*),

$$JT\,[t] = x[t] - |\,acl\,[t] - SMA\,[t]| \quad (3.5)$$

```
// Variable filter akumulasi gerak
double dx1, dy1, da1;
//Filter akumulasi gerak
   dx1 = sum_x * 0.05 ;
   dy1 = sum_y * 0.05 ;
   da1 = sum_a * 0.05 ;
            if (k-2 <= 0){
       dx1 = dx1 ;
       dy1 = dy1 ;
       da1 = da1 ;}
    else{
       dx1 = dx1 + _dx1[k-3]*0.95;
       dy1 = dy1 + _dy1[k-3]*0.95;
       da1 = da1 + _da1[k-3]*0.95;}
```

Gambar 3.18 Intruksi program filter akumulasi gerak

Hasil kalkulasi tersebut kemudian memasuki proses filter nilai gerak getar (*jitter*). Pada metode yang dikembangkan oleh Jinhai Chai dan Rodnet walker [32], proses filter nilai gerak getar menggunakan persamaan yakni sebagai berikut:

$$Js\,[t] = 0.95 * Js\,[t-1] + acl[t] - SMA[t] \tag{3.6}$$

Pada persamaan (3.6) digunakan parameter perbandingan sebesar 0.95. Penggunaan metode tersebut dapat memberikan hasil yang efektif hingga rentang data tertentu. Untuk menyesuaikan dengan kebutuhan sistem stabilisasi video, digunakan metode *Simple Low Pass Filter* dengan *Infinite Impulse Respons* (IIR). Berikut persamaan yang digunakan dalam untuk kompenssasi gerak getar,

$$Js\,[t] = 0.5 * Js\,[t] + 0.5 * JT\,[t] \tag{3.7}$$

Instruksi program untuk proses kalkulasi dan filter gerak getar dapat dilihat pada Gambar 3.19. Keseluruhan instruksi program untuk tahap kompensasi gerak dapat dilihat pada Lampiran A-2.

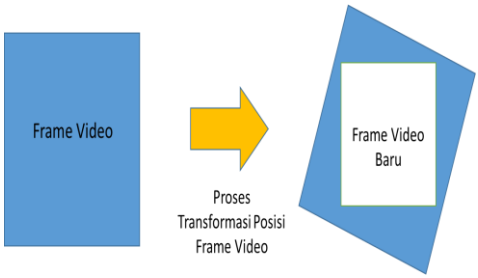
```
//Selisih akumulasi gerak, filter akumulasi
gerak
double pos_x = sum_x - dx1;
double pos_y = sum_y - dy1;
double pos_a = sum_a - da1;

// Kalkulasi gerak getar
double x1_moved = dx + pos_x;
double y1_moved = dy + pos_y;
double a1_moved = da + pos_a;

// Filter gerak getar
x1_moved = x1_moved *0.5;
y1_moved = y1_moved *0.5;
a1_moved = a1_moved *0.5;

if (k-2 <=0){
    x1_moved = x1_moved;
    y1_moved = y1_moved;
    a1_moved = a1_moved;}
else{
    x1_moved = x1_moved + _x1[k-3]*0.5;
    y1_moved = y1_moved + _y1[k-3]*0.5;
    a1_moved = a1_moved + _a1[k-3]*0.5;}

// array gerak getar
 _x1.append(x1_moved);
_y1.append(y1_moved);
_a1.append(a1_moved);
```

Gambar 3.19 Intruksi program kalkulasi dan filter gerak getar

3.2.1.5. *Image Warping*

Pada tahap *image warping* memuat proses transformasi *frame* video. Pertama, dilakukan proses perubahan posisi *frame* video berdasarkan parameter kompensasi gerak. Parameter tersebut diperoleh

dari hasil kalkulasi dan filter gerak getar pada tampilan *frame* video. Proses ini menerapkan metode *affine transform* dengan bantuan fungsi yang terdapat pada *library* OpenCv yakni "warpAffine()". Kemudian, data *frame* video dicuplik sebagian daerahnya untuk *frame* video yang baru. Metode ini digunakan untuk memberikan tampilan video yang ideal. Gambaranumum dari keseluruhan proses pengolahan dapat dilihat pada Gambar 3.20.

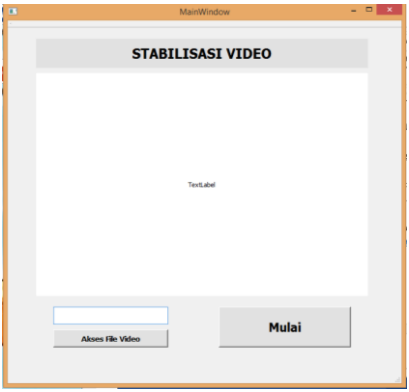


Gambar 3.20 Proses *image warping*

3.3. Desain *Graphical ser Interface*

Sistem stabilisasi video turut dilengkapi dengan *Graphical User Interface* untuk memberikan akses lebih kepada pengguna. Pengguna dapat mengamati hasil proses stabilisasi video serta memberikan data file video yang diolah. Penambahan GUI pada sistem stabilisasi video berperan sebagai penghubung antara sistem dan pengguna terutama dalam proses pertukaran data. Tampilan GUI yang dirancang dapat dilihat pada Gambar 3.21.

Pada Gambar 3.21 terdapat dua fitur utama dalam GUI tersebut yakni fitur "akses file video" dan "Mulai". Diagram alir dari keseluruhan kerja sistem pada GUI dapat dilihat pada Gambar 3.22. Fitur "Akses File Video" memberikan akses pada pengguna untuk memberikan data file video yang ingin diolah. Fitur tersebut memuat intsruksi untuk proses pembacaan data *frame* video serta proses penurunan resolusi gambar. Proses pembacaan data file video menggunakan bantuan *library* OpenCV dengan fungsi "*VideoCapture*". Ketika pembacaan pertama, data *frame* video diinisialisasi sebagai variabel *previous frame* (*frame* sebelumnya). Pada pembacaan berikutnya, data *frame* video dinisialisasi sebagai *current frame* (*frame* saat ini).

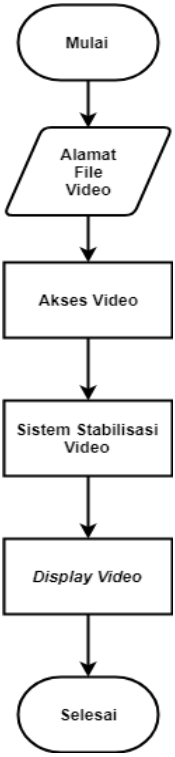


Gambar 3.21 GUI sistem stabilisasi video

Kedua data tersebut digunakan sebagai bahan analisa pada sistem stabillisasi video. Setelah proses analisa selesai, maka data *current frame* diinisialisasi sebagai data *previous frame.* Proses ini dilakukan berulang kali selama data *frame* video dapat terbaca. Proses akses file video dan pembacaan data *frame* video menggunakan instruksi 3seperti pada Gambar 3.23

Sebelum memasuki proses analisa, dilakukan proses penentuan *Region of Interenst* (ROI) dari *frame* video. Proses ini mengambil area tertentu dari data *frame* video untuk diolah lebih lanjut. Hal ini dikarenakan titik piksel yang berlokasi di area tepi sebuah *frame* video memiliki probabilitas tinggi untuk tidak muncul dalam *frame* video berikutnya, terutama saat UAV bergerak [13]. Melalui proses ini, sistem dapat menghemat durasi waktu pengolahan data gambar pada tahap berikutnya. Hasil akhir dari proses ini berupa area gambar yang ideal untuk proses deteksi dan analisa gangguan pada tampilan video.

Region of Interest (ROI) berupa gambar yang memuat cuplikan area *frame* video berbentuk kotak dengan perbandingan ukuran tertentu. Pada fungsi stabilisasi video, ROI memuat 50 persen dari keseluruhan area *frame* video. Daerah yang dicuplik terdiri dari sebagian besar area tengah *frame* video. Daerah tersebut dianggap paling ideal untuk mendeteksi pergerakan objek yang terdapat pada *frame* video. Untuk memisahkan area ROI dengan *frame* video digunakan instruksi seperti pada Gambar 3.24.



Gambar 3.22 Diagram alir GUI

```
VideoCapture cap;
String alamat ("alamat file video");
cap.open(alamat); // akses file video
Mat cur;            // current frame
cap.read(cur);      // read frame
```

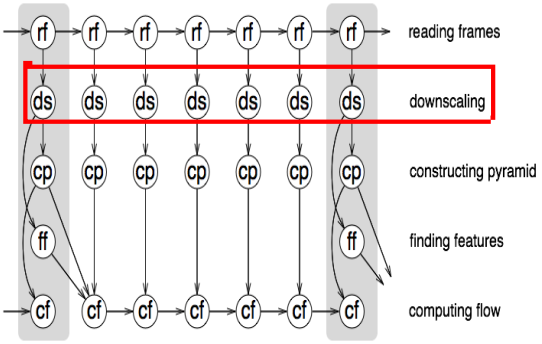
Gambar 3.23 Intsruksi untuk aksess file video

```
Rect kotak1 = Rect (posisi_width,
             posisi_height,
             ukr_width,ukr_height);
Mat PotonganCur = cur (kotak1);   //
```

Gambar 3.24 Instruksi untuk mendapatkan data ROI *frame* video

Data gambar ROI kemudian melalui proses penurunan resolusi gambar pada skala tertentu (*downscale*). Proses ini merupakan bagian dari metode optimalisasi proses stabilisasi video yang dikembangkan oleh Serafim Chekalkin [31]. Menurut Serafing Chekalkin, pada kenyataannya *full frame* video memberikan lebih banyak *noise* daripada dara untuk proses stabilisasi [31]. Keseluruhan metode optimalisasi pada tahap ini dapat dilihat pada Gambar 3.25.

Proses perubahan resolusi gambar menggunakan bantuan *library* OpenCV dengan fungsi "cv::resize()". Instruksi yang digunakan dalam tahap ini dapat dilihat pada Gambar 3.26. Keseluruhan program pada tahap akses file video selengkapnya dapat dilihat pada Lampiran A-1. Hasil akhir dari pengolahan data gambar pada tahap ini digunakan untuk proses deteksi gerak pada tampilan video.



Gambar 3.25 Proses optimalisasi

```
Mat PotonganCur_pro;
Double scale = 0.5;
Resize (PotonganCur, PotonganCur_pro,
       Size(PotonganCur.cols* scale,
       PotonganCur.rows*scale),0,0,
       CV_INTER_LINEAR);
```

Gambar 3.26 Instruksi program untuk menurunkan resolusi ukuran *frame* video

Sedangkan, fitur "Mulai" memberikan akses pada pengguna untuk menjalankan sekaligus menghentikan proses pengolahan data *frame* video. Hasil akhir dari proses pengolahan data *frame* video melalui komunikasi *signal* dan *slots* ditampilkan pada *QLabel*. Keseluruhan instruksi pada GUI dapat dilihat pada Lampiran A-3.

3.4. Desain Modul Uji Performa Sistem Stabilisasi Video

Proses pengujian performa sistem stabilisasi video meliputi dua parameter utama yakni durasi waktu pengolahan (*time processing)* dan kualitas gambar yang dihasilkan (*Image Quality*). Modul uji performa yang dikembangkan berupa *Graphical User Interface* (GUI) yang memuat keseluruhan tahap dan parameter pengujian sistem. Modul uji performa dirangcang untuk mampu menganalisa performa sistem dan kualitas dari hasil pengolahan gambar. Berdasarkan kebutuhan dari proses pengujian performa sistem stabilisasi video, modul uji performa dilengkapi dengan dua mode utama yakni,

- Mode analisa durasi waktu
- Mode analisa kualitas gambar

Kedua mode tersebut memiliki karakteristik yang berbeda dalam proses eksekusinya. Selain kedua mode tersebut, modul uji turut dilengkapi dengan tiga *sub class* program antara lain,

- *Class read video*
- *Class stabilizer*
- *Class test performance*

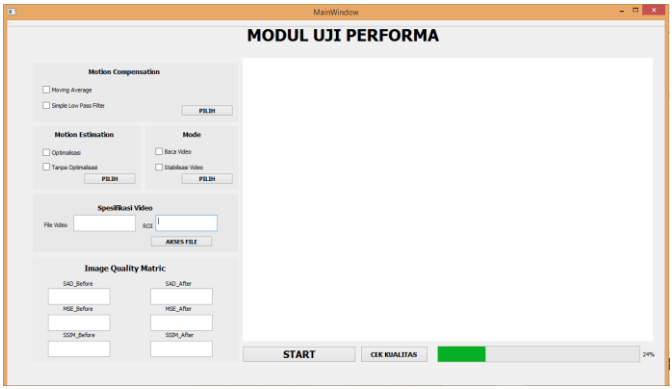
Setiap *class* memiliki fungsi yang berbeda-beda. Pada penerapannya, pemelihan mode sistem menentukan *class* yang dieksekusi oleh sistem. Keseluruhan proses pengolahan data dalam modul uji performa dapat dilihat pada Gambar 3.27.

Pada gambar 3.27 dapat dilihat bahwa mode "Analisa durasi waktu" digunakan untuk mengeksekusi program pada *class read video* dan *class stabilizer*. Sedangkan, pada mode "Analisa kualitas gambar" digunakan untuk mengeksekusi *class test performance.* Berikut spesifikasi dari modul uji performa yang tengah dikembangkan,

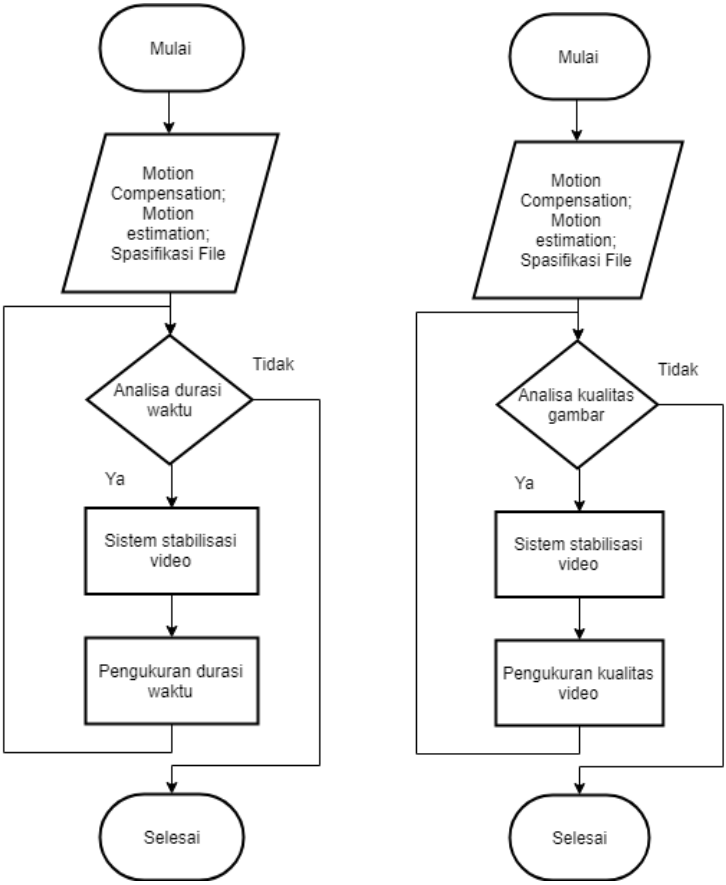
3.4.1. Desain *Graphical User Interface*

Desain *Graphical User Interface* (GUI) menggunakan beberapa *widget* yang tersedia dalam *framework* Qt. Jeni *widget* yang digunakan antara lain *QPushbutton, QLabel, TextEdit,* dan *QProgressbar*. Desain dari tampilan *Graphical user interface* (GUI) dapat dilihat pada Gambar 3.28. Pada gambar 3.28 dapat dilihat bahwa modul uji turut dilengkapi dengan beberapa fitur yang mewakili beberapa metode pengolahan gambar. Keberadaan fitur tersebut memberikan kemudahan bagi pengguna dalam proses pengujian. Hasil pengolahan data *frame* video ditampilkan melalui *QLabel*. Sedangkan, hasil analisa kualitas gambar ditampilkan pada bagian "*Image Quality Matric"*.

Progres dari analisa kualitas gambar ditunjukkan melalui tampilan *QProgressbar*. Keseluruhan instruksi program dapat dilihat pada Lampiran A-3.Data hasil pengujian sistem stabilisasi video secara keseluruhan disimpan dalam bentuk file .txt. Hal ini mempermudah proses analisa performa sistem secara keseluruhan.



Gambar 3.28 GUI modul uji performa



Gambar 3.27 Diagram alir *Graphical User Interface* (GUI) modul uji performa, (a) Analisa durasi waktu (b) Analisa kualitas gambar

3.4.2. Fitur Pendukung pada Modul Uji Performa

Pada modul uji performa sistem terdapat beberapa fitur pendukung yakni antara lain:

- Metode Motion Compensation
- Metode Motion Estimation

- Spesifikasi File Video
- Mode Operasi

Keempat fitur tersebut memberikan kondisi yang berbeda pada proses pengolahan data *frame* video. Kondisi yang diberikan merupakan bagian dari rangkaian pengujian performa fungsi stabilisasi video. Berikut spesifikasi dari keempat mode tersebut,

3.4.2.1. Metode *Motion Compensation*

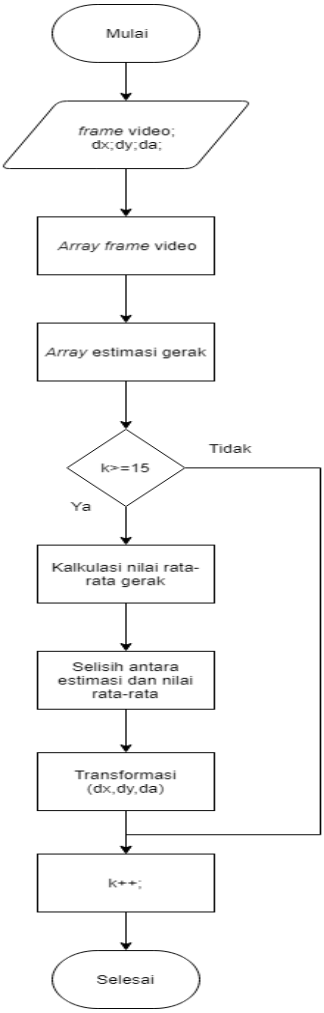
Fitur ini memuat dua jenis metode kompensasi gerak yang berbeda. Fitur metode motion compensation memberikan keleluasaan pada pengguna untuk memilih metode kompensasi gerak yang digunakan. Pada metode kompensasi gerak "*moving average",* untuk memperoleh data transformasi gerak digunakan perbandingan nilai akumulasi gerak dengan rentang posisi hingga 28 *frame* video dari posisi *frame* video saat ini. Rentang *frame* video tersebut terdiri dari 14 *frame* sebelum dan 14 *frame* sesudah posisi saat ini. Diagram alir proses pengolahan data menggunakan metode *moving average* dapat dilihat pada Gambar 3.29.

Sedangkan, pada metode kompensasi gerak dengan "*simple low pass filter"* berdasarkan *Infinite Impulse Response* (IIR) diagram alir proses pengolahan data dapat dilihat pada Gambar 3.30.

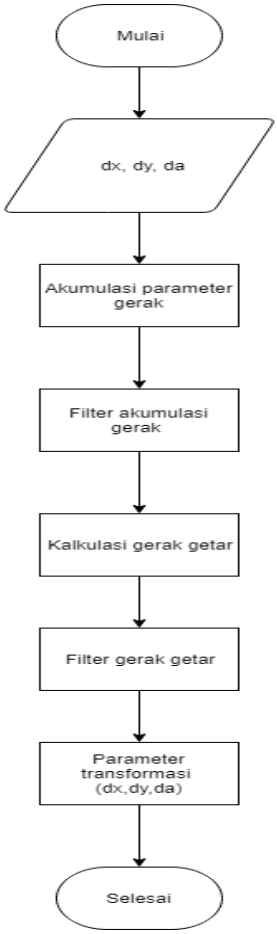
3.4.2.2. Metode *Motion Estimation*

Fitur ini memuat metode untuk memperoleh nilai estimasi gerak *(motion estimation)* dari objek dalam tampilan frame video. Sebelum memasuki tahap *motion estimation* terdapat beberapa tahap pengolahan data frame video yakni tahap *Image pre-processing* dan *Image matching*. Fitur "Metode Motion Estimation" memuat dua kondisi perlakukan berbeda terhadap kedua tahap tersebut. Dua kondisi tersebut diinisialisasi sebagai kondisi "Optimalisasi" dan "Tanpa Optimalisasi".

Setiap kondisi memiliki metode atau teknik yang berbeda dalam mengolah data frame video pada tahap *Image pre-processing* dan *Image matching*. Pada kondisi "Optimalisasi", proses pengolahan data frame video dilakukan dengan menerapkan metode yang dikembangkan oleh Jinhai Chai dan Rodnet walker [32]. Sedangkan, pada kondisi "Tanpa Optimalisasi" proses pengolahan data frame video dapat dilihat pada Gambar 3.31.



Gambar 3.29 Diagram alir metode *Moving average*



Gambar 3.30 Diagram alir metode kompensasi gerak dengan SPLF (IIR)

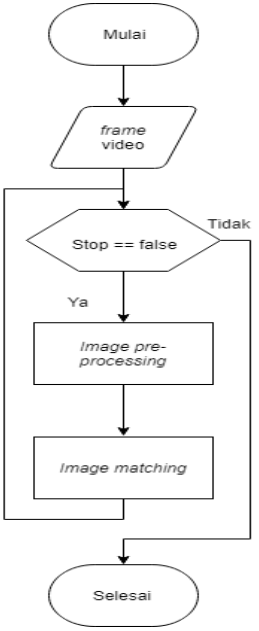
3.4.2.3. Spesifikasi File Video

Fitur ini menyediakan akses peda pengguna untuk menentukan data file video beserta ukuran *Region of Interest* (ROI) untuk proses stabilisasi video. Data file video diakses melalui alamat penyimpanan data tersebut. Fitur ini turut memberikan akses pada sistem stabilisasi

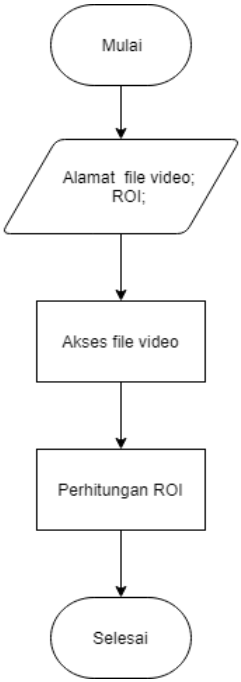
video terhadap beberapa informasi dari file video antara lain ukuran *frame* video hingga nilai *framerate* (FPS). Diagram alir dari keseluruhan proses pengolahan data video pada fitur ini dapat dilihat pada Gambar 3.32.

3.4.2.4. Mode Operasi

Fitur ini menyediakan akses pada pengguna untuk memilih mode operasi yang diinginkan. Mode operasi yang disediakan antara lain "Baca Video" dan "Stabilisasi Video". Kedua mode ini merupakan kondisi pembanding pada mode pengujian utama sistem "Analisa Durasi Waktu". Pada mode "Baca Video", sistem menjalankan *class read video* untuk menganalisa durasi waktu yang diperlukan pada proses pembacaan file video. Sedangkan, pada mode "Stabilisasi Video" sistem menjalankan *class stabilizer* untuk menganalisa durasi waktu yang diperlukan pada proses stabilisasi video.



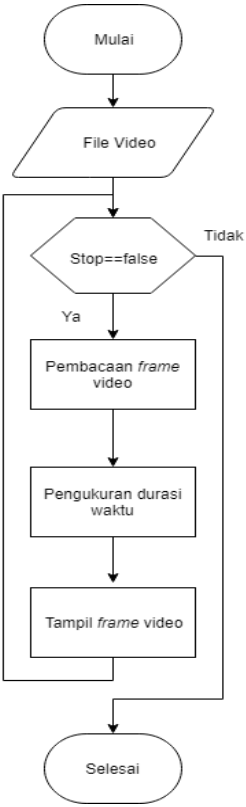
Gambar 3.31 Diagram alir proses pengolahan data pada kondisi "Tanpa optimalisasi"



Gambar 3.32 Diagram alir proses pengolahan data pada fitur "Spesifikasi file video"

3.4.2.5. *Class Read Video*

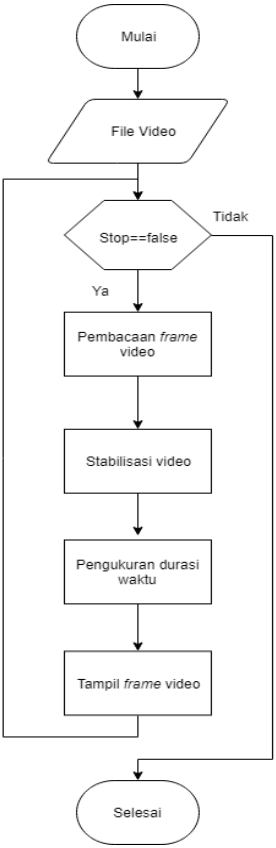
Pada *class* ini memuat proses pembacaan file video dan dilengkapi dengan proses pengukuran durasi waktu. Keseluruhan proses pengolahan data dapat dilihat pada Gambar 3.33. Untuk mengukur durasi waktu, digunakan fungsi "QCurrentDateTime" yang terdapat dalam *library* Qt. Proses pengukuran diawali dengan mendapatkan data durasi waktu pada awal dan akhir proses pengolahan data. Selanjutnya, nilai durasi waktu diperoleh dari selisih antara kedua nilai tersebut. Data durasi waktu yang diberikan dalam satuan *miliseconds* (ms). Hasil akhir dari proses tersebut disimpan dalam file dengan format .txt. Keseluruhan instruksi program dapat dilihat pada Lampiran A-4.



Gambar 3.33 Diagram alir proses pengolahan data *class read video*

3.4.2.6. *Class Stabilizer*

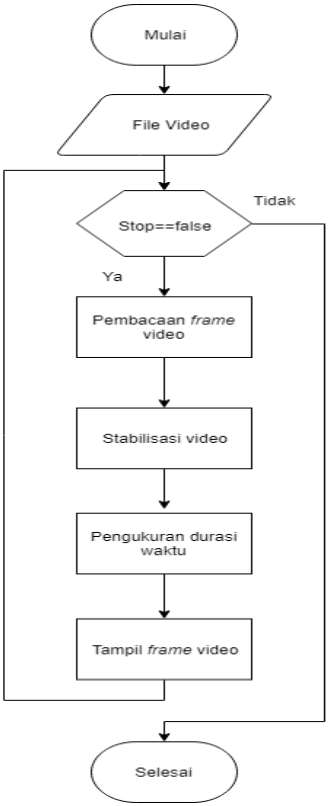
Pada *class* ini memuat keseluruhan tahap stabilisasi video dan proses pengukutan durasi waktu. Keseluruhan proses pengolahan data dapat dilihat pada Gambar 3.34. Proses pengukuran durasi waktu menggunaka prinsip serupa seperti pada *class read video.* Hasil akhir dari proses tersebut disimpan dalam file dengan format .txt. Keseluruhan instruksi program dapat dilihat pada Lampiran A-5.



Gambar 3.34 Diagram alir proses pengolahan data *class stabilizer*

3.4.2.7. *Class Test Performance*

Pada *class* ini memuat keseluruhan tahap stabilisasi video, pengukuran durasi waktu setiap tahap pengolahan data, penyimpanan data nilai estimasi dan transformasi *frame* video, analisa kualitas *motion estimation* serta analisa kualitas video. Keseluruhan proses pengolahan data *frame* video dapat dilihat pada Gambar 3.35. Proses analisa kualitas *motion estimation* berupa pengukuran kualitas kesamaan struktur gambar antara *previous frame* yang telah ditrasformasi dan *current frame*.



Gambar 3.35 Diagram alir proses pengolahan data *class test performance*

Proses ini bertujuan untuk memberikan parameter terhadap kesesuaian hasil estimasi gerak antara dua *frame* video yang saling berurutan. Parameter yang digunakan pada proses ini adalah *Sturctural Similarity Index Matric* (SSIM). Sedangkan, pada proses analisa kualitas video dilakukan pengukuran parameter *Image Quality Matric* antara *previous frame* dan *current frame.* Parameter *Image Quality Matric* terdiri dari nilai *Sum of Absolut Difference* (SAD), *Mean Squared Error*

(MSE) dan *Structural Similarity Index Matric* (SSIM). Keseluruhan instruksi program dapat dilihat pada Lampiran A-6.

3.5. Kriteria Pengujian

Pengujian sistem stabilisasi video menggunakan perangkat *personal computer* (PC) dengan spesifikasi yakni sebagai berikut:

- Tipe : Lenovo G40
- RAM : 2GB DDR3
- Processor : Intel Celeron N2840 -2.1 Ghz
- *Operating System* (OS) : Windows 8.1

Pada proses pengujian digunakan modul uji performa untuk mengukur parameter yang diinginkan. Pengujian menggunakan data file video dengan spesifikasi yakni sebagai berikut:

- Resolusi : 1280 x 720
- *Framerate* (FPS) : 30 frame/s
- Durasi : 3 Menit

Tampilan dari salah satu *frame* video yang diolah dapat dilihat pada Gambar 3.6.

Gambar 3.36 Tampilan *frame* video uji

Pengujian ini berfokus untuk mendapatkan data kualitas proses estimasi gerak serta peforma sistem stabilisasi video. Kedua data tersebut digunakan untuk meninjau kualitas sistem stabilisasi video yang dikembangkan. Kualitas proses estimasi gerak berperan penting dalam keberhasilan metode stabilisasi video yang diterapkan. Kualitas proses

estimasi gerak yang buruk dapat membebani keseluruhan sistem stabilisasi video.

BAB IV
PENGUJIAN DAN ANALISIS DATA

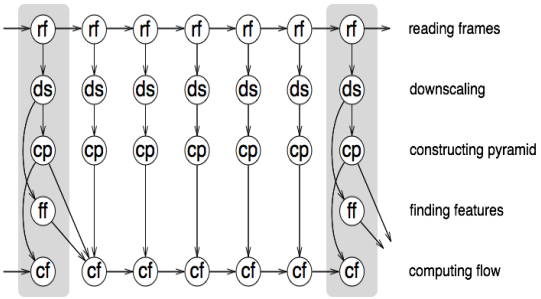
Bab ini memuat kesuluruhan hasil pengujian sistem stabilisasi video beserta dengan analisis data. Kualitas sistem stabilisasi video ditinjau dari kualitas proses estimasi gerak dan performa sistem stabilisasi video. Proses pengujian dilakukan menggunakan modul uji performa. Terdapat dua parameter utama yang dipertimbangkan dalam proses pengujian yakni durasi proses dan tingkat kesamaan gambar.

4.1. Pengujian Kualitas Proses Estimasi Gerak

Kualitas proses estimasi gerak ditinjau dari durasi proses pengolahan data serta tingkat keakuratan hasil estimasi gerak. Pengujian parameter durasi proses pengolahan data menggunakan instruksi "QCurrentDateTime ( )". Intruksi tersebut memberikan akses untuk memperoleh data waktu dari perangkat uji. Sedangkan, pengujian tingkat keakuratan hasil estimasi gerak menggunakan salah satu parameter pada *Image Quality Metric*. Pada pengujian ini digunakan dua kondisi berbeda pada tahap *Image pre-processing* dan *Image matching* yakni kondisi "Optimalisasi" dan "Tanpa Optimalisasi". Pemberian perilaku yang berbeda pada kedua tahap tersebut mempengaruhi keseluruhan proses estimasi gerak.

Kondisi "Optimalisasi" memuat metode optimalisasi yang dikembangkan oleh Serafim Chekalkin [31]. Gambaran umum proses pengolahan data dapat dilihat pada Gambar 4.1. Pada sistem stabilisasi video yang dikembangkan turut menerapkan metode ini. Hal ini dilakukan untuk menghemat durasi pengolahan data tanpa mengurangi tingkat keakuratan hasil estimasi gerak.

Pada Gambar 4.1 dapat dilihat bahwa terdapat modifikasi pada beberapa proses pengolahan data antara lain proses *feature point detection, image pyramid* dan *optical flow matching*. Ketiga proses tersebut memiliki durasi pengolahan paling lama. Pada kondisi "Optimalisasi" ketiga proses tersebut hanya diterapkan pada periode tertentu pengolahan *frame* video. Sedangkan, pada kondisi "Tanpa Optimalisasi" penggunaan proses *feature point detection, image pyramid* dan *optical flow matching* dilakukan pada setiap pengolahan *frame* video.



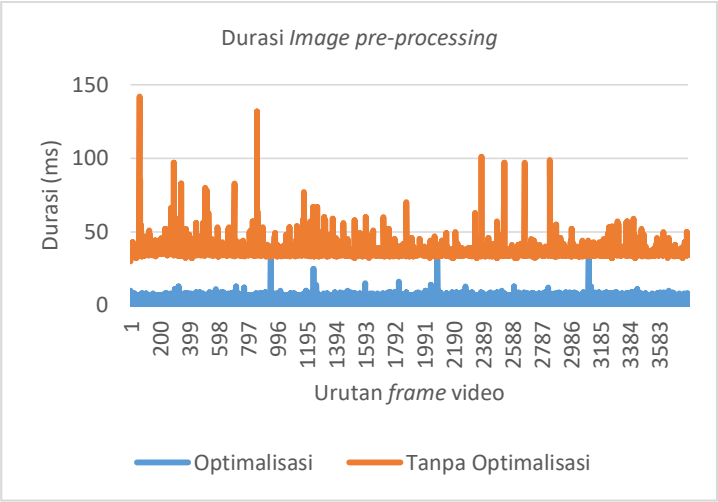
Gambar 4.1 Metode Optimalisasi pada tahap *Image gpre-processing* dan tahap *Image matching* [31]

Pengujian diawali dengan pengukuran parameter durasi pengolahan data *frame* video dengan pemberian kondisi yang berbeda. Pada proses pengukuran parameter ini data yang diolah dalam satuan milisekon (ms). Pengujian dilakukan dengan mengamati durasi pengolahan data *frame* video pada tahap *image pre-processing* dan *image matching*. Hasil pengujian parameter durasi proses pengolahan data *frame* video dapat dilihat pada Tabel 4.1.

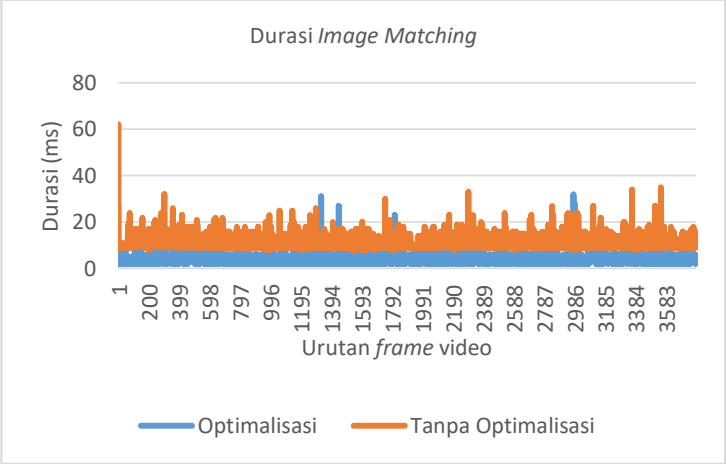
Tabel 4.1 Durasi proses rata-rata

Kondisi	Durasi proses rata-rata (ms)	
	Tahap *Image pre processing*	Tahap *Image Matching*
Optimalisasi	1,77	4,32
Tanpa Optimalisasi	37,14	7,81

Data grafik perbandingan durasi proses pengolahan data pada tahap *image pre-processing* dapat dilihat pada Gambar 4.2. Sedangkan, data grafik perbandingan durasi proses pengolahan pada tahap *image matching* dapat dilihat pada Gambar 4.3.



Gambar 4.2 Grafik durasi proses pengolahan data pada tahap *Image pre processing*

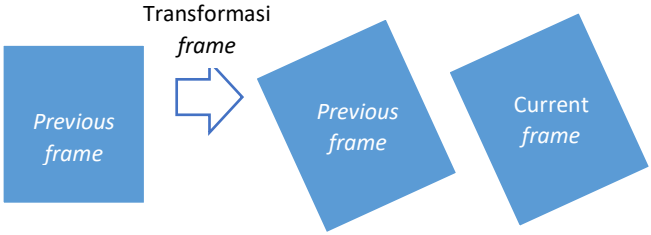


Gambar 4.3 Durasi proses pengolahan data pada tahap *Image Matching*

Berdasarkan Tabel 4.1 diketahui bahwa penggunaan kondisi "Optimalisasi"mampu mengurangi durasi pengolahan data pada tahap

i*mage pre-processing* dan *image matching*. Hal ini didukung dengan data grafik durasi dari kedua tahap tersebut. Berdasarkan Gambar 4.2 dan Gambar 4.3 durasi pengolahan data per *frame* video menunjukkan penurunan durasi yang cukup signifikan ketika penerapan kondisi "Optimalisasi".

Pengujian dilanjutkan dengan mengukur kualitas hasil estimasi gerak oleh sistem stabilisasi video. Kualitas hasil estimasi gerak menunjukkan kemampuan sistem dalam memperhitungkan perubahan posisi objek antara dua *frame* video yang saling berurutan. Pengujian dilakukan dengan membandingkan tingkat kemiripan struktur gambar antara *previous frame* video yang telah ditransformasi berdasarkan hasil estimasi gerak dan *current frame* video. Gambaran umum dari teknik pengujian kualitas hasil estimasi gerak dpat dilihat pada Gambar 4.4.



Gambar 4.4 Gambaran umum pengujian kualitas hasil estimasi gerak

Parameter yang digunakan untuk mengukur tingkat kesamaan struktur berupa *Structural Similarity Index Metric* (SSIM). Parameter tersebut memiliki tingkat akurasi yang cukup tinggi. Hasil pengujian kualitas hasil estimasi gerak pada kedua kondisi dapat dilihat pada tabel 4.2.

Tabel 4.2 Nilai rata-rata SSIM

Kondisi	Nilai rata-rata SSIM
Optimalisasi	0,799
Tanpa Optimalisasi	0,798

Berdasarkan hasil pengukuran pada Tabel 4.2 dapat dilihat bahwa penerapan kondisi "Optimalisasi" memberikan kualitas hasil

estimasi gerak yang tidak jauh berbeda dari kondisi "Tanpa Optimalisai". Sehingga, dapat dikatakan bahwa penerapan metode optimalisasi mampu mengurangi durasi pengolahan data tanpa mengurangi tingkat keakuratan hasil estimasi gerak.

Penggunaan metode optimalisasi memungkinkan sistem untuk menggunakan hasil pemetaan *feature point* pada proses *Optical flow matching* dalam memperkirakan perubahan posisi objek. Selama berlangsungnya proses ini, sistem secara tidak langsung telah mendapatkan data posisi *feature point* untuk *frame* berikutnya. Sehingga, proses deteksi *feature point* dapat dihilangkan. Selain itu, pada proses *Optical flow matching* turut dilakukan effisiensi dengan memangkas adanya penggunaan proses yang berulang.

Proses estimasi gerak turut dipengaruhi oleh beberapa faktor. Salah satunya adalah tampilan objek dalam gambar. Pada video uji, objek utama berupa panorama hutan dan pantai. Kedua jenis panorama tersebut seringkali dihadapi oleh UAV dalam menjalankan misi pengintaian target. Selain itu, gangguan yang dihadapi turut mempengaruhi proses estimasi gerak. Sangat penting untuk memiliki kualitas estimasi gerak yang stabil selama proses pengolahan video.Oleh karena itu, dilakukan pengamatan kualitas hasil estimasi gerak pada rentang durasi video tertentu.

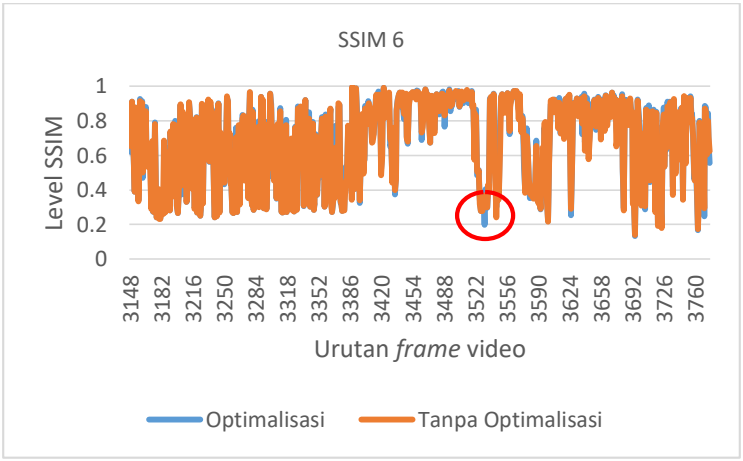
Keseluruhan data pengujian dibagi menjadi kelompok data tertentu pada setiap rentang durasi 30 detik. Perilaku dari setiap kelompok data ditampilkan dalam bentuk grafik tingkat kesamaan struktur. Setiap kelompok durasi data disebut dengan partisi. Total terdapat enam partisi yang masing-masing mewakili data pada tiap rentang durasi 30 detik. Keseluruhan grafik tingkat kesamaan struktur pada kondisi "Optimalisasi" dan "Tanpa Optimalisasi" dapat dilihat di lampiran B-1. Selanjutnya, dilakukan pengukuran nilai rata-rata parameter SSIM pada setiap partisi video. Hasil pengukuran kualitas kesamaan struktur dapat dilihat pada Tabel 4.3.

Tabel 4.3 Nilai rata-rata SSIM

Kondisi	Partisi					
	1	2	3	4	5	6
Optimalisasi	0,858	0,889	0,861	0,727	0,781	0,681
Tanpa Optimalisasi	0,857	0,892	0,864	0,722	0,78	0,675

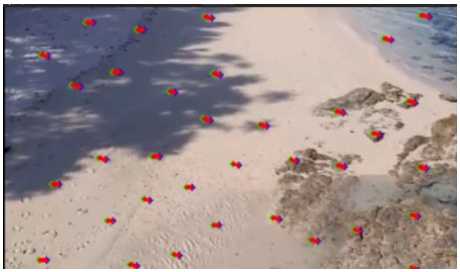
Pada Tabel 4.3 dapat dilihat bahwa terdapat beberapa partisi yang menunjukkan penurunan level kesamaan struktur. Sebagai contoh, pada pertisi ke enam dapat dilihat pada Gambar 4.5. Pada gambar 4.5 penurunan tingkat kesamaan struktur pada daerah yang ditunjukkan oleh lingkaran merah mencapai nilai SSIM sebesar 0.2.

Hal ini menunjukkan ketidaksesuaian hasil estimasi dengan nila pergerakan sebenarnya dari objek. Terdapat beberapa faktor yang menyebabkan kesalahan estimasi gerak. Pada rentang *frame* video partisi 6 dilakukan pengamatan terhadap hasil aliran optik untuk pergerakan objek.



Gambar 4.5 Penurunan tingkat kesamaan struktur pada daerah tertentu (lingkaran merah)

Hasil yang diperoleh berupa adanya perilaku aneh dari aliran optik pada urutan *frame* video tertentu. Kondisi ideal dari aliran optik untuk memberikan estimasi gerak yang sesuai dapat dilihat pada Gambar 4.6.



Gambar 4.6 Aliran optik pada urutan *frame* video ke 3528 (Pada durasi 03.01)

Pada Gambar 4.6 dapat dilihat bahwa aliran optik menunjukkan perubahan normal posisi objek yang terdapat dalam gambar tersebut. Hasil dari aliran optik diinisialisasi sebagai vector gerak untuk proses berikutnya. Namun, dalam proses kalkulasi aliran optik terdapat beberapa gangguan. Kondisi ini menyebabkan aliran optik yang tidak sesuai dengan pergerakan objek dalam gambar. Kondisi tersebut dapat diidentifikasi berdasarkan tampilan aliran optik seperti pada Gambar 4.7.

Gambar 4.7 Aliran optik pada urutan *frame* video ke 3533 (Pada durasi 03.04)

Kondisi tersebut seringkali terjadi ketika sistem dihadapkan dengan perubahan posisi objek yang terlalu besar. Selain itu, sistem juga dapat salah memperkirakan pasangan *feature point* yang koresponden ketika dihadapkan dengan tampilan objek yang hampir serupa. Dampak dari gangguan tersebut berimbas pada hasil estimasi gerak yang

dihasilkan. Hal ini merupakan salah satu kelemahan dari metode *Optical flow matching* yang digunakan. Dalam proses pelacakan *feature point* metode ini memanfaatkan perbedaan intensitas warna. Ketika dihadapkan dengan objek yang hampir mirip dari segi intensitas warna, sistem seringkali salah memperkirakan pasangan *feature point* yang sesuai. Hal yang serupa turut terjadi ketika sistem dihadapkan dengan perubahan posisi *feature point* yang terlalu jauh. Metode *Optical flow* turut didukung dengan konsep *image pyramid* untuk meredam kesalahan estimasi gerak. Namun, tetap terdapat batasan perubahan posisi feature point untuk mendapatkan hasil yang presisi.

4.2. Pengujian Performa Sistem Stabilisasi Video

Performa sistem stabilisasi video dapat ditinjau dari durasi proses pengolahan data *frame* video serta kualitas video yang dihasilkan. Durasi proses pengolahan berdampak pada nilai *framerate* video yang dihasilkan. Sedangkan, kualitas video dapat ditinjau dari tingkat kesamaan struktur gambar antara dua *frame* video yang saling berurutan. Pada pengujian performa sistem stabilisasi video dilakukan pengamatan terhadap pengaruh penerapan kondisi tertentu terhadap performa sistem secara keseluruhan. Hasil pengukuran durasi proses pengolahan data per *frame* video ditampilkan dalam bentuk grafik. Pada pengukuran parameter durasi waktu diterapkan dua kondisi estimasi gerak yakni kondisi "Optimalisasi" dan "Tanpa Optimalisasi".

Hasil pengukuran durasi proses kemudian dibandingkan dengan durasi proses pengolahan data video tanpa adanya proses stabilisasi. Selanjutnya, keseluruhan data durasi proses dibagi menjadi beberapa kelompok partisi. Setiap kelompok partisi mewakili hasil pengukuran tiap durasi 30 detik. Grafik durasi waktu dapat dilihat pada lampiran B-2. Dari hasil pengamatan grafik tersebut, diperoleh nilai rata-rata durasi proses keseluruhan yang dimuat pada Tabel 4.4.

Pada Tabel 4.4 dapat dilihat bahwa pengguanan kondisi "optimalisasi" mempu meningkatkan performa sistem terutama dari segi durasi pengolahan data frame video. Terukur untuk mengolah satu *frame* video memerlukan waktu sekitar 47 milisekon. Dapat dikatakan bahwa kecepeatan proses pengolahan data mencapai rata-rata 21 FPS (*Frame per seconds*). Kecepatan proses pengolahan data *frame* video mendukung untuk diterapkan pada fitur *video streaming*. Penerapan metode "Optimalisasi" mampu mengurangi durasi pengolahan pada tahap *image pre-processing* dan tahap *image matching*. Kedua tahap tersebut memiliki presentase durasi pengolahan paling besar

Tabel 4.4 Rata-rata durasi proses pengolahan *frame* video

Kondisi	Rata-rata durasi proses (ms)
Optimalisasi	47,871
Tanpa Optimalisasi	70,832

Pada pengukuran parameter kualitas video digunakan dua kondisi metode kompensasi gerak yang berbeda yakni metode *Moving average* dan metode kompensasi gerak dengan *Simple low pass filter* (IIR). Parameter *Image quality metric* yang digunakan terdiri dari *Sum of Absolut Difference* (SAD), *Mean Square Error* (MSE) dan *Structural Similarity Index Metric* (SSIM). Ketiga parameter tersebut digunakan untuk mengukur kualitas kesamaan struktur antara dua *frame* yang saling berurutan. Parameter SAD dan MSE seringkali digunakan untuk mengukur kualitas kesamaan struktur dikarenakan komputasinya yang sederhana. Kedua parameter tersebut mengukur tingkat kesamaan struktur berdasarkan perbedaan intensitas warna antara dua *frame* video yang saling berurutan.

Sedangkan, parameter SSIM digunakan untuk mengukur tingkat kesamaan struktur gambar dengan pendekatan *Human Visual System* (HVS). Parameter SSIM mampu mengukur tingkat kesamaan struktur suatu gambar dengan memperhitungkan gangguan berupa *luminance* dan perbedaan kontras cahaya. Pada penerapan metode *moving average* dan metode kompensasi gerak dengan *simple low pass filter (IIR*) dapat memberikan kualitas video yang berbeda. Berikut hasil pengkuran data rata-rata dapat dilihat pada Tabel 4.5.

Tabel 4.5. Hasil pengukuran parameter *Image Quality Matric* sesudah proses stabilisasi

Metode	Parameter	Partisi					
		1	2	3	4	5	6

SPLF (IIR)	SAD	19,576	15,521	16,683	18,492	16,222	19,751
	MSE	1097,308	710,922	769,711	879,133	745,469	999,739
	SSIM	0,457	0,621	0,517	0,406	0,455	0,379
Moving Average	SAD	20,998	18,619	18,225	19,096	18,275	21,328
	MSE	1212,659	952,354	881,179	924,698	886,667	1117,424
	SSIM	0,419	0,511	0,452	0,386	0,389	0,331

Sebagai bahan perbandingan turut dilakukan pengukuran parameter *Image Quality Matric* untuk urutan *frame* video sebelum proses stabilisasi. . Hasil pengukuran dapat dilihat pada Tabel 4.6. Dari kedua data hasil pengukuran diatas, dapat diperoleh data nilai rata-rata dari parameter *Image Quality Metric* baik sebelum maupun sesudah proses stabilisasi video. Data tersebut dapat dilihat pada Tabel 4.7.

Tabel 4.6 Tabel pengukuran parameter *Image Quality Matric* sebelum proses stabilisasi

Parameter	Partisi					
	1	2	3	4	5	6
SAD	21,88	17,94	17,74	19,16	17,53	21,56
MSE	138,71	946,21	889,72	964,21	876,6	1193,7
SSIM	0,429	0,586	0,516	0,408	0,429	0,355

Tabel 4.7 Tabel nilai rata-rata parameter *Image Quality Metric* sebelum dan sesudah proses stabilisasi

Parameter	Nilai rata-rata parameter *Image Quality Metric*	
	Sebelum	Sesudah

		SPLF (IIR)	Moving Average
SAD	19,30	17,71	19,41
MSE	1043,16	867,05	995,83
SSIM	0,454	0,473	0.,15

Dari Tabel 4.7 dapat dilihat bahwa penggunaan metode kompensasi gerak dengan *simple low pass filter* (IIR) memberikan perfoma kualitas video yang lebih baik dibandingkan dengan metode *moving average*. Hal ini ditinjau dari perilaku dari masing-masing parameter kualitas gambar. Penurunan nilai parameter SAD dan MSE atau peningkatan level SSIM menunjukkan struktur gambar yang hampir mirip. Kemampuan sistem dalam meredam efek gerak getar turut dipengaruhi oleh kualitas hasil estimasi gerak. Kualitas hasil estimasi gerak yang rendah dapat mempersulit sistem untuk meredam gangguan pada tampilan video.

Setiap partisi data *frame* video memiliki kualitas hasil estimasi gerak yang berbeda-beda. Partisi ke-6 berdasarkan Tabel 4.3 memiliki kualitas hasil estimasi gerak paling rendah yakni berkisar 0.681. Berdasarkan Tabel 4.5 dan Tabel 4.6 penggunaan *simple low pass filter* (IIR) menunjukkan peningkatan kualitas video pada partisi ke-6. Sedangkan, penggunaan metode *moving average* menunjukkan penurunan kualitas video yang dihasilkan. Berdasarkan pengujian tersebut, terbukti bahwa penggunaan metode kompensasi gerak dengan *simple low pass filter* (IIR) menghasilkan kualitas video yang lebih baik terhadap kondisi gangguan yang beragam. Gangguan tersebut antara lain kondisi objek yang dideteksi hampir mirip satu sama lain, perubahan posisi objek yang terlalu besar hingga perbedaan kontras cahaya.

Penggunaan metode *moving average* dianggap kurang mampu menghadapai adanya perubahan gerak yang tiba-tiba dan cukup besar. Perubahan gerak yang cukup besar dapat berdampak pada parameter transformasi *frame* sebelumnya. Parameter transformasi yang dimaksud berupa nilai rata-rata gerak pada rentang urutan *frame* video tertentu.. Sedangkan, metode *simple low pass filter* (IIR) melalui pendekatan *infinite impulse response* lebih mampu menghadapi berbagai gangguan berupa gerak objek yang tiba-tiba dan cukup besar.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1. Kesimpulan

Pada penelitian ini telah dirancang sistem stabilisasi video untuk mengolah data gambar hasil tangkapan kamera UAV. Sistem stabilisasi video mampu meredam gangguan pada tampilan video. Hal ini ditunjukkan dengan peningkatan kualitas video dengan spesifikasi penurunan parameter SAD sebesar 1,59, MSE sebesar 176,11 dan peningkatan level SSIM sebesar 0,019.

Durasi proses pengolahan data gambar berkisar 47 milisekon untuk mengolah satu *frame* video. Dengan kata lain, kecepatan proses pengolahan gambar berkisar 21 fps (*frame per second*). Kecepatan proses pengolahan gambar tersebut mendukung untuk diterapkan dalam fitur *video streaming* UAV.

5.2. Saran

Berdasarkan hasil penelitian yang telah dilakukan, metode estimasi gerak yang diterapkan seringkali memberikan hasil yang tidak sesuai pada tampilan objek yang hampir serupa maupun perubahan posisi objek yang cukup besar. Sehingga, pada pengembangan berikutnya diperlukan sebuah metode estimasi gerak yang lebih baik.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Hassaballah M., Amin Abdelmgeid Aly, A. Alshazly Hammam." Image Features Detection, Description and Matching". Studies in Computational Intelligence 630, DOI 10.1007/978-3-319-28854-3_2. Springer International Publishing Switzerland .2016

[2] Matsushita Yasuyuki, Eyal, Xiaoou Tang Ofek, Shum Heung-Yeung. "Full-frame Video Stabilization." IEEE Computer Society Conference on Computer Vision and Pattern Recognition .2005

[3] Farima Bento. "Unmanned Aerial Vehicles: An Overview". Potiguese Air Force Academy. January 2008

[4] Singhal Gaurav, Shyam Bansod Babankumar, Mathew Lini. "Unmanned Aerial Vehicle Classification, Applications and Challenges: A Review". Central Scientific Instruments Organization. 2018

[5] E. Torun. "UAV Requirements and design consideration". Turkish Land Forces Command Ankara (Turkey). 2000

[6] C. Soutis. "Fibre-reinforced composites in aircraft construction". Progress in Aerospace Sciences, vol. 41, pp. 143-151. 2005

[7] Rafael C. Gonzales, Richard E. Woods. "Digital Image processing Third Edition". Pearson International Edition prepared by Pearson Education

[8] Akesson Benny, Jansson Henrik. "Video Stabilization for Mobile Equipment". Master of Science in Electrical Engineering Blekinge Institute of Technology. 2003

[9] Rouhafzay Asal. "Video Stabilization Using Point Feature Matching". Master of Science in Electrical and Electronic Engineering. Eastern Mediterranean University. 2015

[10] M. Chethan N.K., Lokesha H. and Ashwarhappa P., "Implementation of a Robust Video Stabilization System on Raspberry PI". International Journal of Innovative Research in Advanced Engineering (IJIRAE). ISSN:2349-2763. 2016.

[11] Q. Zheng, "A Video Stabilization Method Based on Inter-Frame Image Matching Score". Global Journal of Computer Science and Technology: FGraphics & vision. ISSN:0975-4172. 2017

[12] Fleet D, Weiss Y. "Optical flow estimation". In: Handbook of mathematical models in computer vision, Springer, pp 237–257. 2006

[13] Lim Anli, Ramesh Bharath, Yang Yue, Xiang Cheng, Gao Zhi, Lin Feng. “Real-Time Optical flow-based Video Stabilization for Unmanned Aerial Vehicles”. Department of Electrical and Computer Engineering. National University of Singapore. 2017
[14] Bruce D. Lucas. “Image Matching by the Method of Differences”. Carnegie Mellon University. 1984
[15] Patel Dhara, Upadhyay Saurabh. “Optical Flow Measurement using Lucas kanade Method”. 2013
[16] Tarasenko Viacheslav. “Detection and Tracking over Image Pyramids using Lucas and Kanade Algorithm”. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 9 pp 6117-6120. 2016
[17] Dubrofsky Elan. “Homography Estimation”. Master of Science In The Faculty ofGraduate Studies (Computer Science). The University of Britis h Columbia.2009
[18] -. “THE SIMPLEST LOWPASS FILTER”. [online] . Available : https://www.dsprelated.com/freebooks/filters/Simplest_Lowpass_Filter_I.html [Acessed 20 april 2019]
[19] Siemens Phenom. “Introduction to Filters: FIR versus IIR”. [online]. Available : https://community.plm.automation.siemens.com/t5/Testing-Knowledge Base/Introduction-to-Filters-FIR-versus-IIR/ta-p/520959 [Acessed 19 April 2019]
[20] S.B. Balakirsky, R. Chellappa, “Performance characterization of image stabilization algorithms,” Real-Time Imaging, vol. 2, pp. 297–313, 1996
[21] Z. Wang, H. R. Sheikh, and A. C. Bovik, “Objective video quality assessment,” in The Handbook of Video Databases: Design and Applications, B. Furth and O. Marqure, Eds., chapter 41, pp. 1041–1078. CRC Press, 2003
[22] C.Sasi varnan, A.Jagan, Jaspreet Kaur, Divya Jyoti, Dr.D.S.Rao. “Image Quality Assessment Techniques pn Spatial Domain”. IJCST Vol. 2, Issue 3. 2011
[23] R. E. Jacobson. An evaluation of image quality metrics. J. Photograph. Sci., 43:7–16, 1995
[24] Pedersen. Marius. “Image quality metrics for the evaluation of printing workflow”. Faculty of Mathematics and Natural Sciences, University of Oslo No. 1124 ISSN 1501-7710.2011

[25] Banitalebi Amin, Nader-Esfahani Said, Nasiri Avanaki Alireza.” *A Perceptual Based Motion Compensation Technique for Video Coding*”. School of Electrical and Computer Engineering University of Tehran
[26] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image quality assessment: from error visibility to structural similarity”. IEEE Transactions on Image Processing, 13(4):600–612, 2004
[27] Frieyadie. “Panduan Pemograman C++” .penerbit andi: Yogyakarta. 2006
[28] M. Glasser. “Open Verification Methodology Cookbook”. DOI: 10.1007/978-1-4419-0968-8_2. Mentor Graphics Corporation.2009
[29] -. “Qt for Beginners” [online]. Available : https://wiki.qt.io/Qt_for_Beginners [Acessed 28 April 2019]
[30] Nain Neeta, Laxmi Vijay and Bhadviya Bhavitavya. “Feature Point Detection for Real Time Applications”. 2008.
[31] Serafim Chekalkin. “Faster Than Real Time Stabilization of Smartphones Videos”. [online].Available : http://blog.denivip.ru/index.php/2016/04/faster-than-real-time-stabilization-of-videos/?lang=en [Acessed 24 april 2019]
[32] Chai Jinhai, Walker Rodney. “Robust Video Stabilization Algorithm using Feature Point Selection and Delta Optical Flow”. IET Computer Vision, Vol.3, No.4, pp. 176-188. 2009
[33] Niskanen Matti, Olli Silven, Marius Tico.”VIDEO STABILIZATION PERFORMANCE ASSESSMENT”. 2006
[34] Nghiaho. “UNDERSTANDING OPENCV CV::ESTIMATERIGIDTRANSFORM”. [online] . Available : https://nghiaho.com/?p=2208 [Acessed 20 april 2019]
[35] -. "Shi-Tomasi Corner Detector & Good Features to Track". [online]. Available:https://docs.opencv.org/3.0-beta/doc/py_tutorials /py_feature2d/py_shi_tomasi/py_shi_tomasi.html [Accessed 18 April 2019]
[36] BGP. "UAV FIXEDWING VULTURE 130 GNSS". [online]. Available: https://biruni.co.id/product/uav-fixedwing-vulture-130-gnss/ [Accessed 20 April 2019]
[37] ICARUS. "Unmanned Aerial system". [online]. Available: https://icarus.upc.edu/en/research/unmanned-aerial-systems-uas [Acessed 20 April 2019]

[38] Synced. "OpenCV 4.0 Release Ends 3.5 Year Wait".[online]. Available: https://medium.com/syncedreview/opencv-4-0-release-ends-3-5-year-wait-6f3619d156f7 [Accessed 19 April 2019]

LAMPIRAN

A-1 Program*Class* Akses Video dan Transformasi *Frame* Video

Stabilizer.h:

```
  #ifndef STABILIZER_H
  #define STABILIZER_H
  #include <QObject>
  #include <QWidget>
  #include "process_stabilizer.h"
  #include <QThread>
  #include <QtCore>
  #include <QWaitCondition>
  #include <opencv2/core/core.hpp>
  #include <opencv2/imgproc/imgproc.hpp>
  #include <opencv2/highgui/highgui.hpp>
  #include <QPaintEvent>
  #include <iostream>
  #include <fstream>
  #include <QString>
  #include <string>
  #include <QMutex>

  using namespace std;
  using namespace cv;

  class Stabilizer : public QThread
  {
    Q_OBJECT
public:
    Stabilizer(QObject *parent =0);
    ~Stabilizer();
    void play();
    void stop();
    bool Acces_Camera (QString alamat);
    bool isStopped() const;
signals:
      void ResultofStabilization (const QImage &img)
private:
      Process_Stabilizer proces;
      bool Stop;
      QMutex mutex;
      QWaitCondition condition;
      VideoCapture cap;
      QImage img;
      Mat frame;
      double scale = 0.5;
      // Condition of frame
      Mat cur            ;
      Mat prev           ;
      int k=1            ;
      Rect kotak1        ;
      Mat Potongan       ;
      Mat PotonganCur    ;
     Mat Potongan_pro   ;
      Mat PotonganCur_pro;
```

```
      Mat new_frame      ;
      // Specification of ROI
      int ROI_NewFrame   ;
      int posisi_width   ;
      int posisi_height  ;
      int ukr_width      ;
      int ukr_height     ;
      // Specification of video file
       int frame_width     ;
       int frame_height    ;
       int frameRate       ;
       int delay           ;
protected:
       void run();
       void msleep(int ms);
};
#endif // STABILIZER_H

Stabilizer.cpp:
  #include "stabilizer.h"

  #include <QThread>
  #include <QtCore>
  #include <QObject>
  #include <QString>
  #include <QPainter>
  #include <opencv2/opencv.hpp>
  #include <opencv2/core/core.hpp>
  #include <opencv2/imgproc/imgproc.hpp>
  #include <opencv2/highgui/highgui.hpp>
  #include <QDebug>
  #include <iostream>
  #include <fstream>
  #include <string>
  #include <QCheckBox>


  #include <QMessageBox>
  #include <QMutex>
  #include <QTime>

  #include <QDate>
  #include <cassert>
  #include <cmath>

  using namespace std;
  using namespace cv;

  //    For further analysis
        ofstream Sum_frame_perSeconds_pro ("sum_per_seconds_pro.txt");

  Stabilizer::Stabilizer(QObject *parent) : QThread (parent)
  {
    Stop = true;
  }
  Stabilizer::~Stabilizer(){
      mutex.lock();// mengunci kepemiliki atasnya sistem
      Stop =true;
      cap.release(); // melepaskan fungsi VideoCapture
      condition.wakeOne(); // membangan kondisi tunggu untuk beralih ke fungsi lain
```

```
    mutex.unlock(); // melepaskan kepemilikan atasnya
    wait();
}

bool Stabilizer::Acces_Camera(QString alamat){
    string alamat_1 = alamat.toLocal8Bit().data();
    cap.open(alamat_1);
    frame_width = cap.get(CV_CAP_PROP_FRAME_WIDTH);
    frame_height= cap.get(CV_CAP_PROP_FRAME_HEIGHT);
    frameRate   = 30;  delay =(1000/frameRate);

    // for area stabilizer
    posisi_width  = 50 * 0.005 * frame_width;
    posisi_height = 50 * 0.005 * frame_height;
    ukr_width     = 50 * 0.01  * frame_width;
    ukr_height    = 50 * 0.01  * frame_height;}
void Stabilizer::run(){
while (Stop==false){
    if(!cap.read(frame)){
        Stop=true;
        break;
        cap.release();}
    else{
        if (k==1){
           frame.copyTo(prev);
           kotak1    = Rect (posisi_width,posisi_height,ukr_width, ukr_height);
           cout << "position0:" << cap.get(CV_CAP_PROP_POS_FRAMES)<< endl;}
        els{
            frame.copyTo(cur);
          if(!cur.empty()){
            //prev frame
            Potongan     = prev(kotak1)   ;
            resize(Potongan,Potongan_pro,Size(Potongan.cols* scale,Potongan.rows*
                  scale),0,0,CV_INTER_LINEAR);
            //cur frame
            PotonganCur     = cur (kotak1)          ;
            resize(PotonganCur,PotonganCur_pro,Size(PotonganCur.cols*
                  scale,PotonganCur.rows* scale),0,0,CV_INTER_LINEAR);
                //Stabilization Frame
                Mat imgfix =proces.Frame_processing(PotonganCur_pro,Potongan_pro,
                             cur,prev,k);
                 Mat cur22; // Hasil Transformasi
             // Transform frame
                warpAffine(cur, cur22, imgfix, cur.size());
              // Resize curr22
                resize(cur22,cur22,cur.size());
                cur22 = cur22(kotak1);
                  cvtColor(cur22,cur22,CV_BGR2RGB);
                  QVector<QRgb> colorTable;
                  for (int i=0; i<256; i++)
                      colorTable.push_back(qRgb(i,i,i));
                     const unsigned char *qImageBuffer= (const unsigned
                           char*)cur22.data;
                      QImage gambarku=QImage(qImageBuffer,cur22.cols,cur22.rows,
                            cur22.step,QImage::Format_RGB888);
                      gambarku.setColorTable (colorTable);
                      img =gambarku;
                      emit ResultofStabilization(img);
                      this->msleep(delay);}
```

```
                cur.copyTo(prev}
        k++; }}}
void Stabilizer::play(){
    if (!isRunning()){
        if(isStopped()){
            Stop=false;}
        start(LowPriority);}}
void Stabilizer::stop(){
    Stop = true;}
bool Stabilizer::isStopped() const{
    return this->Stop;}
void Stabilizer::msleep(int ms){
    struct timespec ts = {ms/1000,(ms % 1000)*1000*1000};
    nanosleep(&ts,NULL); // pause system}
```

A-2 Program *Class* Pengolahan *Frame* Video

```
Proces_Stabilizer.cpp:
#include "process_stabilizer.h"
#include <QString>
#include <QPainter>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <QDebug>
#include <iostream>
#include <fstream>
#include <string>
#include <QCheckBox>
#include <QMessageBox>
#include <QMutex>
#include <QTime>
#include <QDate>
#include <cassert>
#include <cmath>

using namespace std;
using namespace cv;

Process_Stabilizer::Process_Stabilizer()
{
}
const Mat Process_Stabilizer::Frame_processing( Mat cur, Mat Prev, Mat ori_cur, Mat
ori_prev,int k){
      /********************** frame pre-processing ***********************/
    // vector from prev to cur
    vector <Point2f> prev_corner, cur_corner;
    vector <Point2f> prev_corner2, cur_corner2;
    vector <uchar>   status;
    vector <float>   err;
    vector<Mat> PrevPyramid;
        /*************** Stage 1: Key point extraction*************************/
            if(count ==6){
                count = 1;}
            if(count == 1){
                // Image pre-processing
                cvtColor (Prev,prev_grey,COLOR_BGR2GRAY); // cvt Prev frame
                cvtColor  (cur, cur_grey, COLOR_BGR2GRAY); // cvt cur frame
                //detect key pointc
                 goodFeaturesToTrack(prev_grey, prev_corner, 100, 0.01,
           30,noArray(),5,false,0.04);// detct corner}
            else{
                //Image pre-processing
                 cvtColor  (cur, cur_grey, COLOR_BGR2GRAY); // cvt cur frame
                for(size_t i=0; i < move_corner.size(); i++){
                    prev_corner.push_back(move_corner[i]); }}
             count ++;
            /****************Stage 2: Image Matching**************************/
             if(prev_corner.size() > 0){
```

```
                if(k==2){
                   buildOpticalFlowPyramid(prev_grey,PrevPyramid,Size(21,21),3,true);
                    buildOpticalFlowPyramid(cur_grey,CurPyramid, Size(21,21),3,true);
                    //Estimasi aliran optik
                     calcOpticalFlowPyrLK( PrevPyramid,CurPyramid, prev_corner,
                             cur_corner,status, err,Size(21,21),3);//Optical Flow
                   Matching}
                else{
                    if(ak ==0){
                      buildOpticalFlowPyramid(cur_grey,Pyramidd,Size(21,21),3,true);
                      //Estimasi aliran optik
                      calcOpticalFlowPyrLK(CurPyramid, Pyramidd, prev_corner,
                   cur_corner,status, err,Size(21,21),3);// Optical Flow Matching
                      ak++;}
                    else{
                     buildOpticalFlowPyramid(cur_grey,CurPyramid,Size(21,21),3,true);
                      calcOpticalFlowPyrLK(Pyramidd, CurPyramid,
                        prev_orner,cur_corner,status, err,Size(21,21),3);
                       ak =0;}}
                // weed out bad matches
                for(size_t i=0; i < status.size(); i++) {
                    if(status[i]) {
                        prev_corner2.push_back(prev_corner[i]);
                        cur_corner2.push_back(cur_corner[i]);}}
                             //Save_move_corner
                        move_corner.clear();
                        for(size_t i=0; i <cur_corner2.size(); i++){
                             move_corner.push_back(cur_corner2[i]);
                         }
                         // Condition Match Result
                         Match_Con =1;}
           else{
                Match_Con =0; // Condition Match Result}

       /****************** Stage 3: Motion Estimation ******************/
      if(prev_corner2.size()){
          if(Match_Con > 0){
    T = estimateRigidTransform(prev_corner2, cur_corner2, false);
    if(T.data == NULL) {
                    last_T.copyTo(T);}
              T.copyTo(last_T);
             // decompose T
              dx = T.at<double>(0,2);
              dy = T.at<double>(1,2);
              da = atan2(T.at<double>(1,0), T.at<double>(0,0));// sudut}
         else{
             dx =0; dy =0;da =0;}}
      else{
           dx =0; dy =0;da =0;}
      //Nilai estimasi sebenarnya untuk proses translasi
      dx = dx/scale;
      dy = dy/scale;
/******************* Stage 4: Motion Compensation******************/
```

```
    Mat T1_new (2,3,CV_64F);
    T1_new = Compensation_Method(dx,dy,da,k);
    return T1_new;}
const Mat Process_Stabilizer::Compensation_Method(double dx, double dy, double
da, int k)
{
 // for final matrix transformation
     Mat T1 (2,3,CV_64F);
    _x.append(dx); _y.append(dy); _a.append(da);
    for (int i=-11; i < -1; i++){
         int a= k+i;
         if (a < 0){
             sum_x +=0;
             sum_y +=0;
             sum_a +=0;}
         else{
             sum_x +=_x[a];
             sum_y +=_y[a];
             sum_a +=_a[a]; }}
// Filter motion of the frame
     dx1 = sum_x * 0.05 ;
     dy1 = sum_y * 0.05 ;
     da1 = sum_a * 0.05 ;
   if (k-2 <= 0){
       dx1 = dx1 ;
       dy1 = dy1 ;
       da1 = da1 ;}
  else{
       dx1 = dx1 + _dx1[k-3]*0.95;
       dy1 = dy1 + _dy1[k-3]*0.95;
       da1 = da1 + _da1[k-3]*0.95;}
//Motion_Smoothing
    _dx1.append(dx1); _dy1.append(dy1); _da1.append(da1);
//Motion Compensation
    int pos_x = sum_x - dx1;
    int pos_y = sum_y - dy1;
    int pos_a = sum_a - da1;
     if (pos_x < 0){
         pos_x = pos_x;}
    else{
         pos_x = -pos_x;}
    if (pos_y <0){
         pos_y = pos_y;}
     else{
          pos_y = -pos_y;}
    if (pos_a < 0){
          pos_a = pos_a;}
     else{
          pos_a = -pos_a; }
    x1_moved = dx + pos_x;
    y1_moved = dy + pos_y;
    a1_moved = da + pos_a;

     x1_moved = x1_moved *0.5;
     y1_moved = y1_moved *0.5;
```

```
a1_moved = a1_ moved *0.5;

          if (k-2 <=0){
             x1_moved = x1_moved;
             y1_moved = y1_moved;
             a1_moved = a1_moved;}
          else{
              x1_moved = x1_moved + _x1[k-3]*0.5;
              y1_moved = y1_moved + _y1[k-3]*0.5;
              a1_moved = a1_moved + _a1[k-3]*0.5;}
          _x1.append(x1_moved);_y1.append(y1_moved);_a1.append(a1_moved);
          dx1_new =  x1_moved ;
          dy1_new =  y1_moved ;
          da1_new =  a1_moved ;
           //Clean sum
          sum_x = 0; sum_y = 0; sum_a = 0;
dx1_new = -dx1_new ;
dy1_new = -dy1_new ;
da1_new = -da1_new ;
 T1.at<double>(0,0) = cos(da1_new);
 T1.at<double>(0,1) = -sin(da1_new);
 T1.at<double>(1,0) = sin(da1_new);
 T1.at<double>(1,1) = cos(da1_new);
 T1.at<double>(0,2) = dx1_new;
 T1.at<double>(1,2) = dy1_new;
 return T1;}
```

A-3 Program GUI Sistem Stabilisasi Video

```
MainWindow.h:
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>

#include "stabilizer.h"
using namespace std;
using namespace cv;

namespace Ui {
class MainWindow;}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void ShowImage (QImage img);
    void on_pushButton_8_clicked();
    void on_pushButton_7_clicked();
private:
    Ui::MainWindow *ui;
    Stabilizer *vid_stabil;};
#endif // MAINWINDOW_H

MainWindow.cpp:

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    vid_stabil   = new Stabilizer();
    QObject ::connect(vid_stabil, SIGNAL(ResultofStabilization(QImage)),
                      this,SLOT(ShowImage(QImage)));
    ui->setupUi(this);}

MainWindow::~MainWindow(){
    delete ui;}

// Show The Result
void MainWindow::ShowImage(QImage img){
 ui->label_2->setPixmap(QPixmap::fromImage(img).scaled(ui->label_2->size(),
                Qt::KeepAspectRatio,Qt::FastTransformation));}

// start
void MainWindow::on_pushButton_8_clicked(){
    vid_stabil->play();}
// File
```

```
void MainWindow::on_pushButton_7_clicked(){
    QString alamat =ui->textEdit_2->toPlainText();
     vid_stabil->Acces_Camera(alamat);}
```

A-4 Program *Class Read Video*

read_video.cpp:

```
#include "read_video.h"
#include <QThread>
#include <QtCore>
#include <QObject>
#include <QString>
#include <QPainter>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <QDebug>
#include <iostream>
#include <fstream>
#include <string>
#include <QCheckBox>
#include <QMessageBox>
#include <QMutex>
#include <QTime>
#include <QDate>
#include <cassert>
#include <cmath>

using namespace std;
using namespace cv;

//    For further analysis
      ofstream Sum_frame_perSeconds_pro ("sum_per_seconds_pro.txt");

read_video::Stabilizer(QObject *parent) : QThread (parent)
{
  Stop = true;
}

read_video::~Stabilizer(){
    mutex.lock();// mengunci kepemiliki atasnya sistem
    Stop =true;
    cap.release(); // melepaskan fungsi VideoCapture
    condition.wakeOne(); // membangan kondisi tunggu untuk beralih ke fungsi lain
    mutex.unlock(); // melepaskan kepemilikan atasnya
    wait();

}


bool read_video::Acces_Camera(QString alamat){
    string alamat_1 = alamat.toLocal8Bit().data();
    cap.open(alamat_1);
    frame_width = cap.get(CV_CAP_PROP_FRAME_WIDTH);
    frame_height= cap.get(CV_CAP_PROP_FRAME_HEIGHT);
    frameRate   = 30;  delay =(1000/frameRate);}

void read_video::run(){

while (Stop==false){
double tim1 = get_Time_mSec();
```

```
    if(!cap.read(frame)){
        Stop=true;
        break;
        cap.release();}
    else{
                if(!frame.empty()){
                 frame = frame (kotak1);
                 cvtColor (frame,imgfix_2,CV_BGR2RGB);
                 // Convert to image
                 QVector<QRgb> colorTable;
                 for (int i=0; i<256; i++)
                  colorTable.push_back(qRgb(i,i,i));
                  const unsigned char *qImageBuffer= (const unsigned
char*)imgfix_2.data;
                   QImage gambarku=
QImage(qImageBuffer,imgfix_2.cols,imgfix_2.rows,
                              imgfix_2.step,QImage::Format_RGB888);
                  gambarku.setColorTable (colorTable);
                 img =gambarku;
                 emit ResultofRead_1 (img);
                 // Cek time
                 double tim2 = get_Time_mSec();
                 Time_diff (tim1,tim2);
                 this->msleep(delay);
             }
         }

void read_video::Time_diff(int sec,int sec_1){
    double diff = sec_1 - sec;
    if (diff < 0){
        double dif_1 = 1000- sec;
        diff = dif_1 + sec_1;}
    Sum_frame_perSeconds_pro << diff << endl}
double read_video::get_Time_mSec(){
    QString in =QDateTime::currentDateTime().toString("yyyy/MM/dd hh:mm:ss,
zzz");
    QString time = in.right(3);
    double ok = time.toDouble();
    return ok;}

void read_video::play(){
    if (!isRunning()){
        if(isStopped()){
            Stop=false;}
        start(LowPriority);}}

void read_video::stop(){
    Stop = true;}

bool read_video::isStopped() const{
    return this->Stop;}

void read_video::msleep(int ms){
    struct timespec ts = {ms/1000,(ms % 1000)*1000*1000};
    nanosleep(&ts,NULL); // pause system}
```

A-5 Program *Class Stabilizer*

```
Stabilizer_Function.cpp:
#include "Stabilizer_Function.h"
#include <QThread>
#include <QtCore>
#include <QObject>
#include <QString>
#include <QPainter>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <QDebug>
#include <iostream>
#include <fstream>
#include <string>
#include <QCheckBox>
#include <QMessageBox>
#include <QMutex>
#include <QTime>

#include <QDate>
#include <cassert>
#include <cmath>

using namespace std;
using namespace cv;

//    For further analysis
      ofstream Sum_frame_perSeconds_pro ("sum_per_seconds_pro.txt");

Stabilizer::Stabilizer(QObject *parent) : QThread (parent)
{
  Stop = true;
}

Stabilizer_Function::~Stabilizer_Function (){
    mutex.lock();// mengunci kepemiliki atasnya sistem
    Stop =true;
    cap.release(); // melepaskan fungsi VideoCapture
    condition.wakeOne(); // membangan kondisi tunggu untuk beralih ke fungsi
lain
    mutex.unlock(); // melepaskan kepemilikan atasnya
    wait();
}

bool Stabilizer_Functionr::Acces_Camera(QString alamat){
    string alamat_1 = alamat.toLocal8Bit().data();
    cap.open(alamat_1);
    frame_width = cap.get(CV_CAP_PROP_FRAME_WIDTH);
    frame_height= cap.get(CV_CAP_PROP_FRAME_HEIGHT);
    frameRate   = 30;  delay =(1000/frameRate);

    // for area stabilizer
    posisi_width  = 50 * 0.005 * frame_width;
    posisi_height = 50 * 0.005 * frame_height;
    ukr_width     = 50 * 0.01  * frame_width;
```

```
    ukr_height    = 50 * 0.01  * frame_height;}
void Stabilizer_Function ::run(){
while (Stop==false){
    if(!cap.read(frame)){
        Stop=true;
        break;
        cap.release();}
    else{
        double tim1 = get_Time_mSec();
        if (k==1){
           frame.copyTo(prev);
           kotak1    = Rect (posisi_width,posisi_height,ukr_width, ukr_height);
           cout << "position0:" << cap.get(CV_CAP_PROP_POS_FRAMES)<< endl;}
        els{
            frame.copyTo(cur);
              if(!cur.empty()){
                //prev frame
                Potongan     = prev(kotak1)   ;
                resize(Potongan,Potongan_pro,Size(Potongan.cols*
scale,Potongan.rows*
                              scale),0,0,CV_INTER_LINEAR);
                //cur frame
                PotonganCur     = cur (kotak1)          ;
                resize(PotonganCur,PotonganCur_pro,Size(PotonganCur.cols*
                              scale,PotonganCur.rows*
                              scale),0,0,CV_INTER_LINEAR);
                //Stabilization Frame
              Mat imgfix =proces.Frame_processing(PotonganCur_pro,Potongan_pro,
                         cur,prev,k);
              Mat cur22; // Hasil Transformasi
             // Transform frame
                warpAffine(cur, cur22, imgfix, cur.size());
              // Resize curr22
                resize(cur22,cur22,cur.size());
                cur22 = cur22(kotak1);
                  cvtColor(cur22,cur22,CV_BGR2RGB);
                  QVector<QRgb> colorTable;
                  for (int i=0; i<256; i++)
                      colorTable.push_back(qRgb(i,i,i));
                     const unsigned char *qImageBuffer= (const unsigned
                              char*)cur22.data;
                      QImage gambarku=QImage(qImageBuffer,cur22.cols,cur22.rows,
                                        cur22.step,QImage::Format_RGB888);
                      gambarku.setColorTable (colorTable);
                      img =gambarku;
                      emit ResultofStabilization(img);
                      this->msleep(delay);}
                cur.copyTo(prev}
        double tim2 = get_Time_mSec();
        Time_diff (tim1,tim2);
        k++; }}}

void Stabilizer_Function::Time_diff(int sec,int sec_1){
    double diff = sec_1 - sec;
    if (diff < 0){
        double dif_1 = 1000- sec;
        diff = dif_1 + sec_1;}
    Sum_frame_perSeconds_pro << diff << endl}

double Stabilizer_Function::get_Time_mSec(){
```

```
    QString in =QDateTime::currentDateTime().toString("yyyy/MM/dd hh:mm:ss,
zzz");
    QString time = in.right(3);
    double ok = time.toDouble();
    return ok;}

void Stabilizer_Function::play(){
    if (!isRunning()){
        if(isStopped()){
            Stop=false;}
        start(LowPriority);}}

void Stabilizer_Function::stop(){
    Stop = true;}

bool Stabilizer_Function::isStopped() const{
    return this->Stop;}

void Stabilizer_Function::msleep(int ms){
    struct timespec ts = {ms/1000,(ms % 1000)*1000*1000};
    nanosleep(&ts,NULL); // pause system}
```

A-6 Program *Class Test Performance*

```
#include "test_performance.h"
#include <QThread>
#include <QtCore>
#include <QObject>
#include <QString>
#include <QPainter>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <QDebug>
#include <iostream>
#include <fstream>
#include <string>
#include <QCheckBox>
#include <QMessageBox>
#include <QMutex>
#include <QAxObject>
#include <QTime>

#include <QDate>
#include <cassert>
#include <cmath>
using namespace std;
using namespace cv;

Test_Performance::Test_Performance(QObject *parent) : QThread (parent)
{
    Stop = true;
}
Test_Performance::~Test_Performance(){
    mutex.lock();// mengunci kepemiliki atasnya sistem
    Stop =true;
    cap.release(); // melepaskan fungsi VideoCapture
    condition.wakeOne(); // membangan kondisi tunggu untuk beralih ke fungsi
lain
    mutex.unlock(); // melepaskan kepemilikan atasnya
    wait();
}

// File Penyimpan hasil SAD dan MSE and SSIM
        ofstream SAD_intensity_Bfr ("SAD_intensity_bfr.txt");
        ofstream MSE_intensity_Bfr ("MSE_intensity_bfr.txt");
        ofstream SSIM_Intensity_Bfr ("SSIM_intensity_bfr.txt");
        ofstream SAD_intensity_Afr ("SAD_intensity_afr.txt");
        ofstream MSE_intensity_Afr ("MSE_intensity_afr.txt");
        ofstream SSIM_Intensity_Afr ("SSIM_intensity_afr.txt");

//    For further analysis
        ofstream time_read ("time_read.txt");
        ofstream time_rect ("time_rect.txt");
        ofstream time_Save ("time_save.txt");
        ofstream time_processing_ALL ("tp_ALL.txt");
        ofstream Sum_frame_perSeconds ("sum_per_seconds.txt");
        ofstream Resolution_width ("width.txt");
        ofstream Resolution_height ("height.txt");
```

```
        ofstream Time_SAD_MSE   ("SAD_MSE_time.txt");
        ofstream Time_SSIM   ("SSIM_time.txt");

bool Test_Performance::Acces_Camera(QString Ip, int ROI, int method, int optimal){
    string alamat = Ip.toLocal8Bit().data();
    cap.open(alamat);
    ROI_NewFrame = ROI;
    frame_width = cap.get(CV_CAP_PROP_FRAME_WIDTH);
    frame_height= cap.get(CV_CAP_PROP_FRAME_HEIGHT);
    frameRate   = 30;  delay =(1000/frameRate);
    // prev frame pre-processing
    // for area stabilizer
    int posisi = 100 - ROI_NewFrame;
    posisi_width  = posisi * 0.005 * frame_width;
    posisi_height = posisi * 0.005 * frame_height;
    ukr_width     = ROI_NewFrame * 0.01  * frame_width;
    ukr_height    = ROI_NewFrame * 0.01  * frame_height;

    //for area estimation
    //for area estimation
    posisi_width_pro  = ukr_width    * 0.25;
    posisi_height_pro = ukr_height   * 0.25;
    ukr_width_pro     = ukr_width    * 0.5 ;
    ukr_height_pro    = ukr_height   * 0.5 ;
    process.Num_Method_test(method);
    Method  = method;
    Optimal_ = optimal;
    process.Optimal_test(optimal);
}

double Test_Performance::getNumberOfFrames(){
    return double(cap.get(CV_CAP_PROP_FRAME_COUNT));
}

double Test_Performance::get_SSIM(Mat T1, Mat T2){
    Mat cur_test; cvtColor(T1, cur_test ,CV_BGR2GRAY);
    Mat cur22_test; cvtColor (T2, cur22_test, CV_BGR2GRAY);
    const double C1 = 6.5025, C2 = 58.5225;
        /***************************** INITS **********************************/
        int d = CV_32F;
        Mat I1, I2;
        cur_test.convertTo(I1, d);         // cannot calculate on one byte large
values
        cur22_test.convertTo(I2, d);
        Mat I2_2   = I2.mul(I2);        // I2^2
        Mat I1_2   = I1.mul(I1);        // I1^2
        Mat I1_I2  = I1.mul(I2);        // I1 * I2

        /*************************** END INITS **********************************/
        Mat mu1, mu2;                   // PRELIMINARY COMPUTING
        GaussianBlur(I1, mu1, Size(11, 11), 1.5);
        GaussianBlur(I2, mu2, Size(11, 11), 1.5);
        Mat mu1_2   =   mu1.mul(mu1);
        Mat mu2_2   =   mu2.mul(mu2);
        Mat mu1_mu2 =   mu1.mul(mu2);
        Mat sigma1_2, sigma2_2, sigma12;
        GaussianBlur(I1_2, sigma1_2, Size(11, 11), 1.5);
        sigma1_2 -= mu1_2;
```

```
        GaussianBlur(I2_2, sigma2_2, Size(11, 11), 1.5);
        sigma2_2 -= mu2_2;
        GaussianBlur(I1_I2, sigma12, Size(11, 11), 1.5);
        sigma12 -= mu1_mu2;

        //////////////////////////// FORMULA ////////////////////////////////
        Mat t1, t2, t3;
        t1 = 2 * mu1_mu2 + C1;
        t2 = 2 * sigma12 + C2;
        t3 = t1.mul(t2);              // t3 = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))
        t1 = mu1_2 + mu2_2 + C1;
        t2 = sigma1_2 + sigma2_2 + C2;
        t1 = t1.mul(t2); //t1 =((mu1_2 + mu2_2 + C1)*(sigma1_2 + sigma2_2 + C2))
        Mat ssim_map;
        divide(t3, t1, ssim_map);        // ssim_map =  t3./t1;
        Scalar mssim = mean(ssim_map);
        double MSSIM = sum(mssim)[0];// mssim = average of ssim map
        return MSSIM;
}

double Test_Performance::getSAD(Mat I1, Mat I2){
          cvtColor(I1, cur_test ,CV_BGR2GRAY);
         CvtColor(I2, cur22_test, CV_BGR2GRAY);
         int mean = ukr_height * ukr_width;
         for (int i= 1 ; i < cur_test.cols; i++){
                for(int j = 1; j < cur_test.rows; j++){
                  int intensity1 = (int) cur22_test.at<uchar>(j,i);
                  int intensity2 = (int) cur_test.at<uchar>(j,i)  ;
                  int difference = intensity2-intensity1;
             // for SAD
                  if(difference < 0){
                   jumlah = jumlah - difference;}
                  else {
                   jumlah = jumlah +difference;}}}
             // get SAD vALUE
              double sad = jumlah/mean;
              SAD_Value.append(sad);
              jumlah =0;
              return sad;}

double Test_Performance::getMSE(Mat i1, Mat i2){
    cvtColor(i1, cur_test ,CV_BGR2GRAY);
    cvtColor(i2, cur22_test,CV_BGR2GRAY);
    int mean = ukr_height * ukr_width;
    for (int i= 1 ; i < cur_test.cols; i++){
        for(int j = 1; j < cur_test.rows; j++){
          int intensity1 = (int) cur22_test.at<uchar>(j,i);
          int intensity2 = (int) cur_test.at<uchar>(j,i)  ;
          int difference = intensity2-intensity1;
    // for MSE
          kuadrat = difference;
          kuadrat = kuadrat * kuadrat;
          jumlah_kuadrat += kuadrat;}}
    // get MSE Value
     double mse = jumlah_kuadrat/mean;
     MSE_Value.append(mse);
     jumlah_kuadrat = 0;
     return mse;}
```

```
void Test_Performance::run(){
     std::string lokasivideo= "E:/OpticalFlow.avi";

     video.open(lokasivideo,CV_FOURCC('M','J','P','G'),frameRate,Size(frame_width
     ,frame_height,true);
    std::string loc ="E:/videoStabil.avi"
while (Stop==false){
    int tim1 = get_Time_Sec();
    if(!cap.read(frame))
    {
        Stop=true;
        break;
        cap.release();
        video.release();
        C_video.release();}
    else{
        if (k==1)
        {    frame.copyTo(prev);
           kotak1 = Rect (posisi_width,posisi_height,ukr_width, ukr_height);
           kotak_pro = Rect (posisi_width_pro, posisi_height_pro, ukr_width_pro,
                    ukr_height_pro);
           if(Method == 1){
               // Save_Certain frame
               frame_save [factor_1] = prev;
               imwrite ("prev.png", frame_save [factor_1]);
               factor_1++;}}
        else
        {
            // Read Cur Frame
            Read_tim_1 = get_Time_mSec(); //check Read Time
             frame.copyTo(cur);
            Read_tim_2 = get_Time_mSec();
            time_read << diff_time_msec(Read_tim_1, Read_tim_2)<< endl;// Save
Read time
            if(!cur.empty()){
                if(Method == 1){
                    //Save_Certain frame
                    if(factor_1 == 15){
                        factor_1 = 0; }
                    frame_save[factor_1] = cur;
                    if(k==2){
                        imwrite ("cur.png", cur);}
                     factor_1 ++;}


/**************************Frame_Processing********************/
                   // Mode Optimal untuk motion estimation
                   if(Optimal_ == 1){
                       // Rect prev frame
                          Potongan     = prev(kotak1)   ;
                          resize(Potongan,Potongan_pro,Size(Potongan.cols*
                                       scale,Potongan.rows*
                                   scale),0,0,CV_INTER_LINEAR);
                        //Rect cur frame
                           Rect_tim_1 = get_Time_mSec();
                           PotonganCur     = cur(kotak1)   ;

resize(PotonganCur,PotonganCur_pro,Size(PotonganCur.cols*
```

```
                                    scale,PotonganCur.rows*
                              scale),0,0,CV_INTER_LINEAR);
                    Rect_tim_2 = get_Time_mSec();
                    time_rect << diff_time_msec(Rect_tim_1,Rect_tim_2)<< endl;

                     //Stabilization Frame
                      Pro_tim_1 = get_Time_mSec();// get data time

                    imgfix = process.Frame_processing(PotonganCur_pro,
          Potongan_pro,
                              cur,prev,kotak1,scale,k);// Matriks Transformas
                     Pro_tim_2 = get_Time_mSec();
                  time_processing_ALL << diff_time_msec( Pro_tim_1, Pro_tim_2)
<< endl ;}
                  // Mode biasa untuk motion estimation
                  else{
                      // Rect prev frame
                         Potongan     = prev(kotak1)   ;
                       //Rect cur frame
                          Rect_tim_1 = get_Time_mSec();
                          PotonganCur     = cur(kotak1)   ;
                          Rect_tim_2 = get_Time_mSec();
                          time_rect << diff_time_msec(Rect_tim_1,Rect_tim_2)<<
endl;
                          //Stabilization Frame
                          Pro_tim_1 = get_Time_mSec();// get data time
                          imgfix = process.Frame_processing(PotonganCur,
                                       Potongan,cur,prev,kotak1,scale,k);//
                             Matriks Transformas
                          Pro_tim_2 = get_Time_mSec();
                   time_processing_ALL << diff_time_msec( Pro_tim_1, Pro_tim_2)
<< endl ;
                  }
/***************Check Structure and Save*************************************/
               if(Method == 1){
                   if(k>=15){
                       // return to initial position certain frame
                       if(fact_1 == 15){
                           fact_1 = 0;}
                       Mat cur22; // Hasil Transformasi
                       // Transform frame
                       warpAffine(frame_save[fact_1], cur22, imgfix,cur.size());
                       if (k<=16){
                            prev_new_frame = cur22 (kotak1);
                            frame_save[fact_1](kotak1).copyTo(prev_ori_frame);
                            if (k==16){
                                imwrite ("contoh1.png", frame_save[fact_1]);}}
              else{
                     //Check_STructure_Before
                      //Save  SAD_MSE value
                            sad_1 =
                     getSAD(prev_ori_frame,frame_save[fact_1](kotak1));
                           mse_1 = getMSE(prev_ori_frame,frame_save[fact_1]
                               (kotak1));
                           SAD_intensity_Bfr << sad_1 << " " << endl;
                            MSE_intensity_Bfr << mse_1 << " " << endl;
                      // Save SSIM_Vlue
                           MSSIM_1 = get_SSIM(prev_ori_frame,frame_save[fact_1]
                                      (kotak1));
```

```
                            SSIM_Intensity_Bfr << MSSIM_1 << endl;
                             // Cek Structure After
                              new_frame = cur22 (kotak1);
                              SAD_MSE_tim_1 = get_Time_mSec();

                              //Save  SAD_MSE value
                               sad = getSAD(new_frame,prev_new_frame);
                               mse = getMSE(new_frame,prev_new_frame);
                              SAD_intensity_Afr << sad << " " << endl;
                              MSE_intensity_Afr << mse<< " " << endl;
                              emit ResultofSAD_Value(sad,sad_1);
                              emit ResultofMSE_Value(mse,mse_1);
                              SAD_MSE_tim_2 =get_Time_mSec();
                         Time_SAD_MSE <<diff_time_msec(SAD_MSE_tim_1,SAD_MSE_tim_2)<<
endl;
                              SSIM_tim_1 = get_Time_mSec();

                              // Save SSIM_Value
                               MSSIM = get_SSIM(new_frame,prev_new_frame);
                              SSIM_Intensity_Afr << MSSIM << endl;
                              SSIM_tim_2 = get_Time_mSec();
                              emit ResultofSSIM_Value(MSSIM,MSSIM_1);
                              Time_SSIM <<diff_time_msec(SSIM_tim_1,SSIM_tim_2)<<
endl;
                              Save_tim_1 = get_Time_mSec();
                               new_frame.copyTo(prev_new_frame);
                               frame_save[fact_1](kotak1).copyTo(prev_ori_frame);
                          }

// Video Writer
// Now draw the original and stablised side by side for coolness
Mat Curr; Mat Prevvv;
resize(prev_new_frame,Curr,cur.size()); // stabil frame
resize(frame_save[fact_1](kotak1),Prevvv,cur.size()); // original frame
Mat canvas = Mat::zeros(cur.rows, cur.cols*2+10, cur.type());
Prevvv.copyTo(canvas(Range::all(), Range(0, Curr.cols)));
Curr.copyTo(canvas(Range::all(), Range(Curr.cols+10, Curr.cols*2+10)));

// If too big to fit on the screen, then scale it down by 2, hopefully it'll fit
:)

  if(canvas.cols > 1920) {
    cv::resize(canvas, canvas, Size(canvas.cols/3, canvas.rows/3));}
// Convert to image
   resize(canvas,canvas,cur.size());
   C_video.write(canvas); // Save Stabil
   Mat Optical = process.Copy_frame();
    resize(Optical,Optical,cur.size());
   video.write(Optical); // Save transform
fact_1 ++;}}
  else{
       Mat cur22; // Hasil Transformasi
       // Transform frame
       warpAffine(cur, cur22, imgfix, cur.size());
       if (k==2){
           prev_new_frame = cur22 (kotak1);}
      else{
                        //Check_STructure_Before
                            //Save  SAD_MSE value
```

```
                        sad_1 = getSAD(PotonganCur,Potongan);
                        mse_1 = getMSE(PotonganCur,Potongan);
                        SAD_intensity_Bfr << sad_1 << " " << endl;
                        MSE_intensity_Bfr << mse_1 << " " << endl;
                        if(k ==3){
                            imwrite ("gambar3.png",PotonganCur);
                            imwrite ("gambar4.png",Potongan); }

                       // Save SSIM_Value
                        MSSIM_1 = get_SSIM(PotonganCur,Potongan);
                        SSIM_Intensity_Bfr << MSSIM_1 << endl;
                    //Check_Structure_After
                    new_frame = cur22 (kotak1);
                    SAD_MSE_tim_1 = get_Time_mSec();
                    //Save  SAD_MSE value
                    int sad = getSAD(new_frame,prev_new_frame);
                    int mse = getMSE(new_frame,prev_new_frame);
                    SAD_intensity_Afr << sad << " " << endl;
                    MSE_intensity_Afr << mse<< " " << endl;
                    emit ResultofSAD_Value(sad,sad_1);
                    emit ResultofMSE_Value(mse,mse_1);
                    SAD_MSE_tim_2 =get_Time_mSec();
                    Time_SAD_MSE <<diff_time_msec(SAD_MSE_tim_1,
                          SAD_MSE_tim_2)<< endl;
                    SSIM_tim_1 = get_Time_mSec();

                    // Save SSIM_Value
                    double MSSIM = get_SSIM(new_frame,prev_new_frame);
                    SSIM_Intensity_Afr << MSSIM << endl;
                    SSIM_tim_2 = get_Time_mSec();
                    emit ResultofSSIM_Value(MSSIM,MSSIM_1);
                    Time_SSIM <<diff_time_msec(SSIM_tim_1,SSIM_tim_2)<<
                                endl;
                    Save_tim_1 = get_Time_mSec();
                     prev_new_frame = new_frame;
                    }

                //Video_writer
                // Now draw the original and stablised side by side
                Mat Curr; Mat Prevvv;
                resize(prev_new_frame,Curr,cur.size()); // stabil frame
                resize(PotonganCur,Prevvv,cur.size()); // original frame
                Mat canvas = Mat::zeros(cur.rows, cur.cols*2+10,cur.type());
                Prevvv.copyTo(canvas(Range::all(), Range(0, Curr.cols)));
                Curr.copyTo(canvas(Range::all(), Range(Curr.cols+10,
                                  Curr.cols*2+10)));
                if(canvas.cols > 1920) {
                cv::resize(canvas, canvas, Size(canvas.cols/3,
                           canvas.rows/3));}

                 // Convert to image
                  resize(canvas,canvas,cur.size());
                  C_video.write(canvas); // Save Stabil
                  Mat Optical = process.Copy_frame();
                    resize(Optical,Optical,cur.size());
                    video.write(Optical); // Save transform}}
           cur.copyTo(prev);}}
    int tim2 = get_Time_Sec();
```

```
        Time_diff (tim1,tim2,k);
        emit urutan(k);
        k++;}}

void Test_Performance::Time_diff(int sec,int sec_1, int count){
    int diff = sec_1 - sec;
    if (diff > 0){
        int diff_1 = count - afr;
        Sum_frame_perSeconds << diff_1 << endl;
        afr=count;}}

double Test_Performance::get_Time_mSec(){
    QString in =QDateTime::currentDateTime().toString("yyyy/MM/dd hh:mm:ss, zzz");
    QString time = in.right(3);
    double ok = time.toDouble();
    return ok;}

double Test_Performance::diff_time_msec(double SEC, double SEC_1){
    double diff = SEC_1 - SEC;
    return diff;}

int Test_Performance::get_Time_Sec(){
    t = std::time(0);
    now_S = std::localtime(&t);
    sec = now_S->tm_sec;
    return sec;}

void Test_Performance::play(){
    if (!isRunning()){
        if(isStopped()){
            Stop=false;}
        start(LowPriority);}}

void Test_Performance::stop(){
    Stop = true;
}

bool Test_Performance::isStopped() const{
    return this->Stop;
}

void Test_Performance::msleep(int ms){
    struct timespec ts = {ms/1000,(ms % 1000)*1000*1000};
    nanosleep(&ts,NULL); // pause system
}
```

A-7 Program GUI Uji Sistem Stabilisasi Video

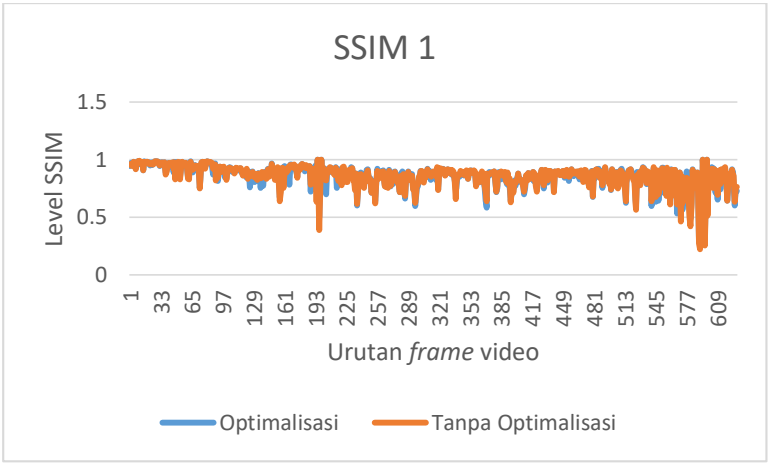
```
MainWindow.h:
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include "stabilizer_function.h"
#include "test_performance.h"
#include "read_video.h"
#include "processing_frame.h"
#include "processing_frame_test.h"
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();
    void on_pushButton_2_clicked();
    void ShowImage (QImage img);
    void ShowRealVideo (QImage img);
    void ShowSAD_Value (double SAD, double SAD_1);
    void ShowMSE_Value (double MSE, double MSE_1);
    void ShowSSIM_Value (double SSIM, double SSIM_1);
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_5_clicked();
    void on_progressBar_valueChanged(int value);
    void on_pushButton_6_clicked();

private:
    Ui::MainWindow *ui;
    Stabilizer_Function *vid_stabil;
    read_video *read;
    Test_Performance *test;
    processing_frame *Process;
    processing_frame_test *Process_test;
    int Method;
    int optimal;
    int kondisi =0;
    int kondisi_1 =0;
    int mode;
};
#endif // MAINWINDOW_H

MainWindow.cpp:

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
```

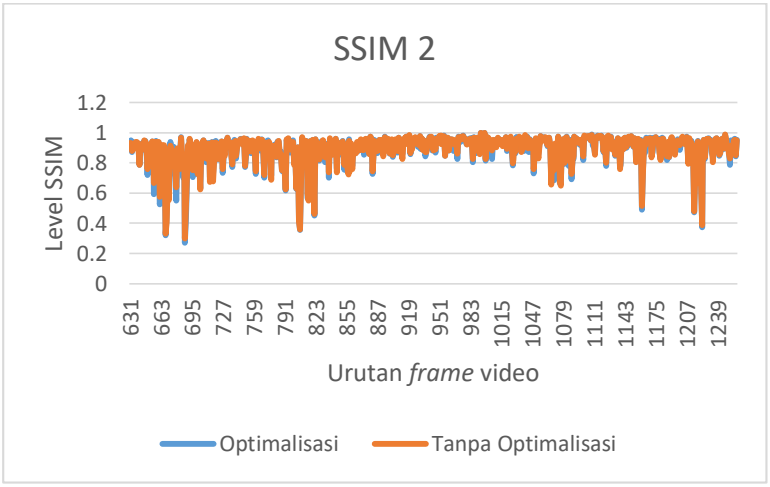
```
{
    vid_stabil   = new Stabilizer_Function();
    read = new read_video();
    test = new Test_Performance();
    QObject ::connect(vid_stabil, SIGNAL(ResultofStabilization(QImage)),
                      this,SLOT(ShowImage(QImage)));
    QObject::connect(test, SIGNAL(ResultofSAD_Value(double, double)),
                      this,SLOT(ShowSAD_Value(double,double)));
    QObject::connect(test,SIGNAL(ResultofMSE_Value(double,double)),
                      this,SLOT(ShowMSE_Value(double,double)));
    QObject::connect(read,SIGNAL( ResultofRead_1(QImage)),
                      this,SLOT(ShowRealVideo(QImage)));
    QObject::connect(test, SIGNAL(ResultofSSIM_Value(double,double)),this,
                    SLOT(ShowSSIM_Value(double,double)));

QObject::connect(test,SIGNAL(urutan(int)),this,SLOT(on_progressBar_valueChanged(
int)));
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

//START
void MainWindow::on_pushButton_clicked()
{
    if(kondisi == 0){
        if(mode ==1){
             read ->play();}
        else{
            vid_stabil ->play();}
        ui->pushButton->setText("STOP");
        kondisi = 1;}
    else{
        if(mode == 1){
            read ->stop();}
        else{
            vid_stabil ->stop();}
        ui->pushButton->setText("START");
        kondisi =0;}}

//Acces_Camera
void MainWindow::on_pushButton_2_clicked()
{
    QString IP =ui->textEdit_5->toPlainText();
    int Ip = IP.toInt();
    QString ROI = ui ->textEdit_2->toPlainText();
    int roi = ROI.toInt();
    vid_stabil->Acces_Camera(IP,roi,Method, optimal);
    read->Acces_camera1(IP,roi);
    test -> Acces_Camera(IP,roi,Method,optimal);
    qDebug() << "Mat" << Method;
    ui->progressBar->setOrientation(Qt::Horizontal);
    ui->progressBar->setValue(0);}

// Show The Result
void MainWindow::ShowImage(QImage img){
```

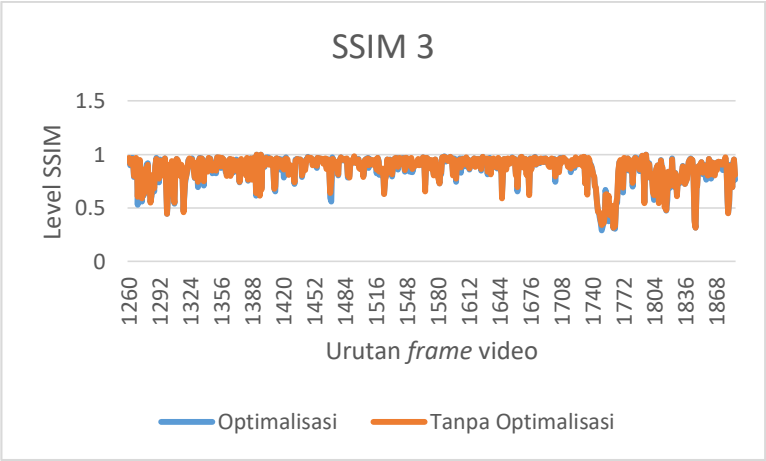
```
ui->label_2->setPixmap(QPixmap::fromImage(img).scaled(ui->label_2->size(),
            Qt::KeepAspectRatio,Qt::FastTransformation));}

// Show real Video
void MainWindow::ShowRealVideo(QImage img){
ui->label_2->setPixmap(QPixmap::fromImage(img).scaled(ui->label_2->size(),
             Qt::KeepAspectRatio,Qt::FastTransformation));}

void MainWindow::ShowSAD_Value(double SAD,double SAD_1){
    QString SAD_New = QString::number(SAD);
    ui->textEdit->setText(SAD_New);
    QString SAD_New_1 = QString::number(SAD_1);
    ui->textEdit_7->setText(SAD_New_1);}

void MainWindow::ShowMSE_Value(double MSE, double MSE_1){
    QString MSE_New = QString::number(MSE);
    ui->textEdit_3->setText(MSE_New);
    QString MSE_New_1 = QString::number(MSE_1);
    ui->textEdit_8->setText(MSE_New_1);
}

void MainWindow::ShowSSIM_Value(double SSIM, double SSIM_1){
    QString SSIM_New = QString::number(SSIM);
    ui->textEdit_4->setText(SSIM_New);
    QString SSIM_New_1 = QString::number(SSIM_1);
    ui->textEdit_6->setText(SSIM_New_1);
}

void MainWindow::on_pushButton_3_clicked()
{
    if(ui->checkBox->isChecked()){
        Method =1;}
    if(ui->checkBox_2->isChecked()){
        Method =2;}}

void MainWindow::on_pushButton_4_clicked()
{
    if(ui->checkBox_3->isChecked()){
        optimal =1;}
    if(ui->checkBox_4->isChecked()){
        optimal =2}}

void MainWindow::on_pushButton_5_clicked()
{
    if(kondisi_1 == 0){
        test ->play();
        ui->pushButton_5->setText("STOP");
        kondisi_1 = 1;}
    else{
        test ->stop();
        ui->pushButton_5->setText("START");
        kondisi_1 =0;}}

void MainWindow::on_progressBar_valueChanged(int value){
    ui->progressBar->setRange(1,(test->getNumberOfFrames()));
    ui->progressBar->setValue(value);
}

void MainWindow::on_pushButton_6_clicked()
```

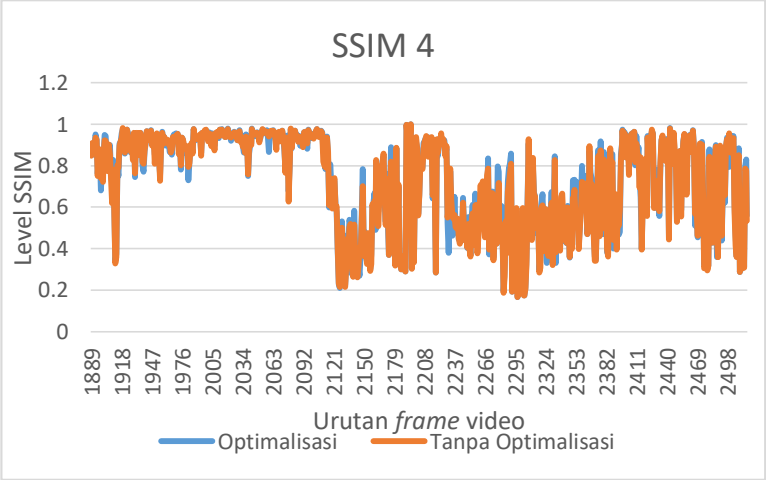
```
{
    if(ui->checkBox_5->isChecked()){
        mode =1;
    }
    if(ui->checkBox_6->isChecked()){
        mode =2;
    }
}
```

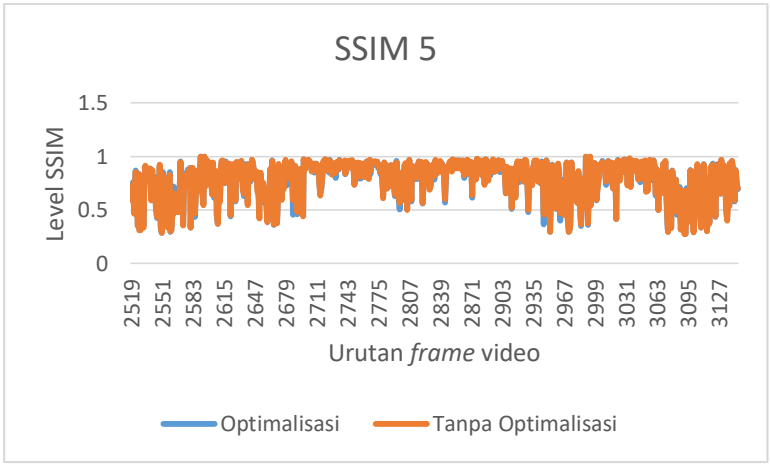
B-1 Hasil Pengujian Kualitas Hasil Estimasi Gerak

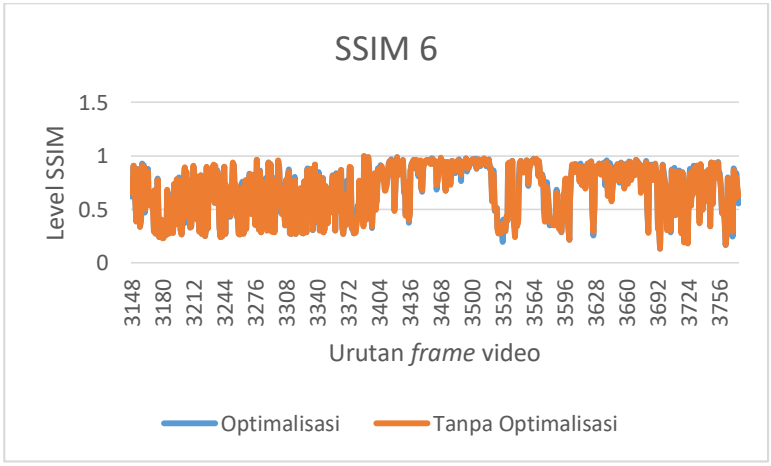




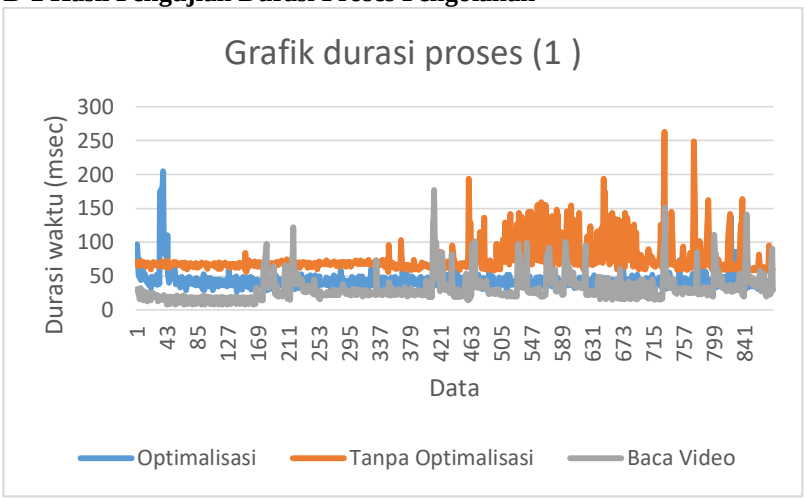

SSIM 3
Level SSIM
1.5
1
0.5
0
1260
1292
1324
1356
1388
1420
1452
1484
1516
1548
1580
1612
1644
1676
1708
1740
1772
1804
1836
1868
Urutan *frame* video
Optimalisasi
Tanpa Optimalisasi

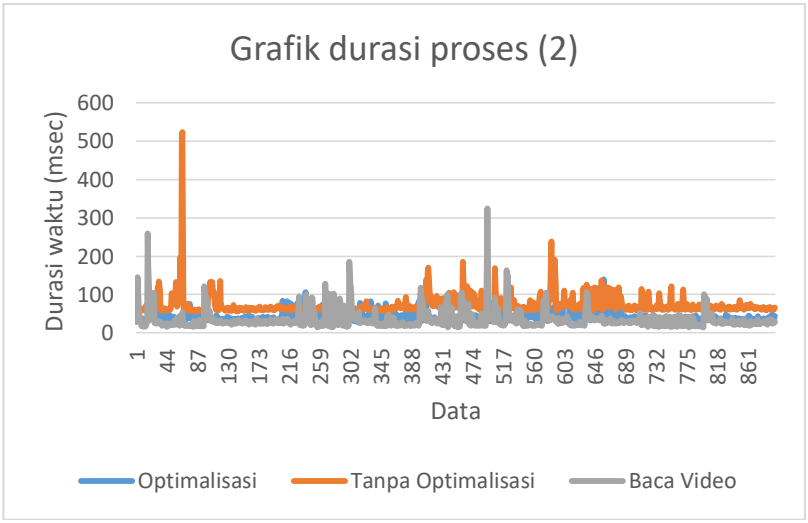

SSIM 4
Level SSIM
1.2
1
0.8
0.6
0.4
0.2
0
1889
1918
1947
1976
2005
2034
2063
2092
2121
2150
2179
2208
2237
2266
2295
2324
2353
2382
2411
2440
2469
2498
Urutan *frame* video
Optimalisasi
Tanpa Optimalisasi

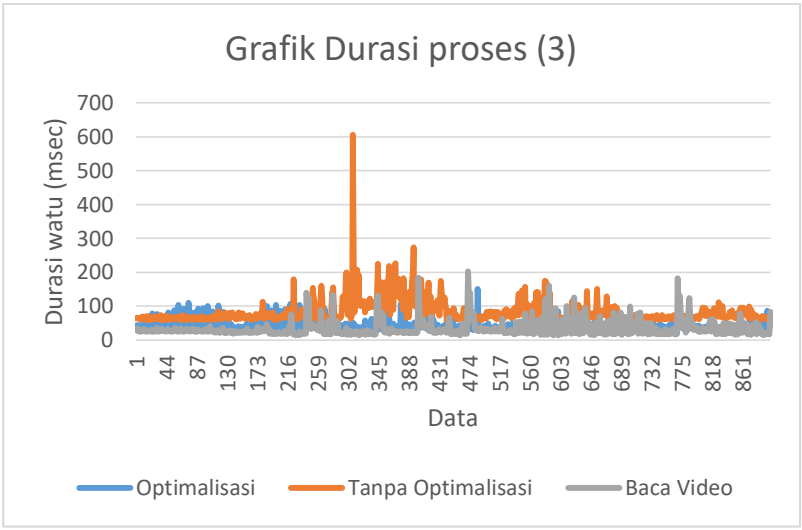

SSIM 5
Level SSIM
1.5
1
0.5
0
2519
2551
2583
2615
2647
2679
2711
2743
2775
2807
2839
2871
2903
2935
2967
2999
3031
3063
3095
3127
Urutan *frame* video
Optimalisasi
Tanpa Optimalisasi

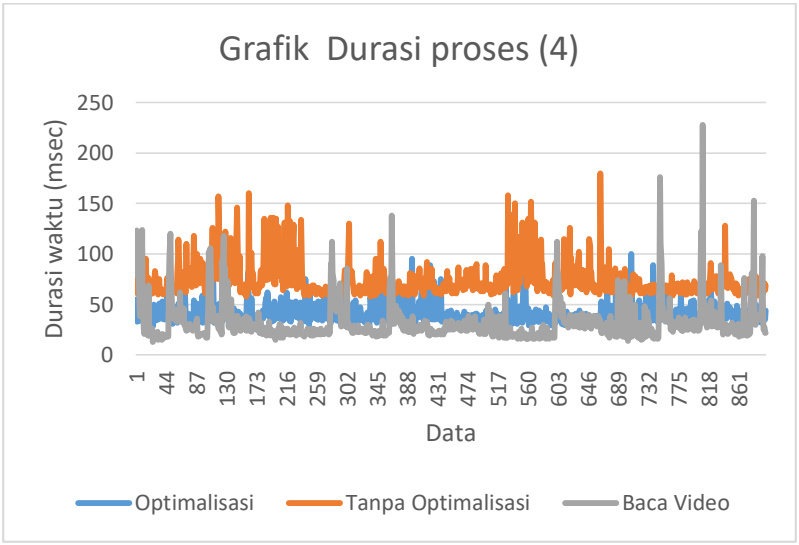

SSIM 6
Level SSIM
1.5
1
0.5
0
3148
3180
3212
3244
3276
3308
3340
3372
3404
3436
3468
3500
3532
3564
3596
3628
3660
3692
3724
3756
Urutan *frame* video
Optimalisasi
Tanpa Optimalisasi

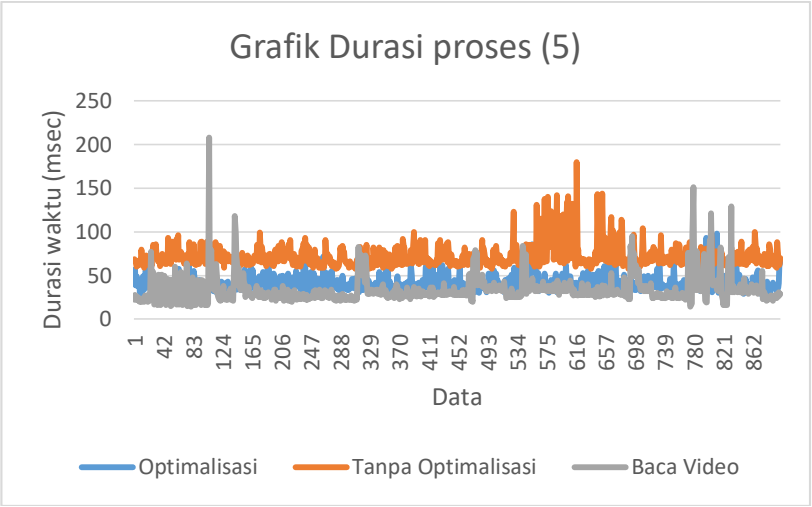
B-2 Hasil Pengujian Durasi Proses Pengolahan

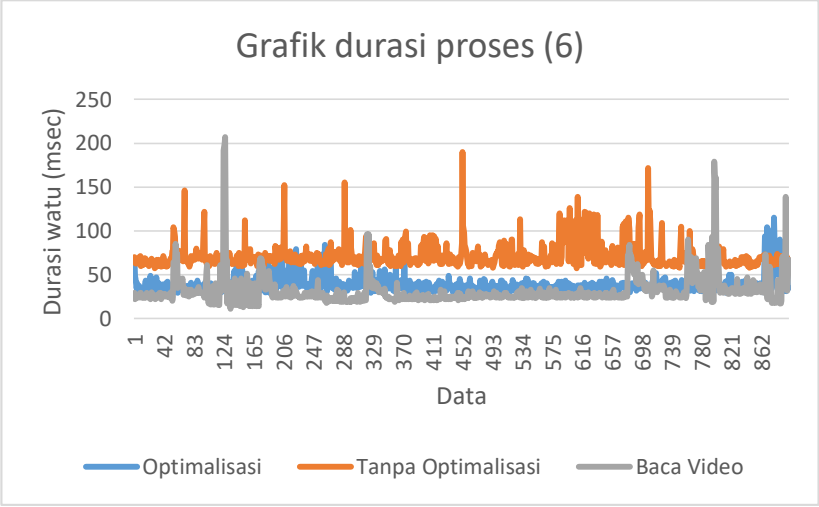





Grafik Durasi proses (3)
Durasi watu (msec)
700
600
500
400
300
200
100
0
1
44
87
130
173
216
259
302
345
388
431
474
517
560
603
646
689
732
775
818
861
Data
Optimalisasi
Tanpa Optimalisasi
Baca Video


Grafik Durasi proses (4)
Durasi waktu (msec)
250
200
150
100
50
0
1
44
87
130
173
216
259
302
345
388
431
474
517
560
603
646
689
732
775
818
861
Data
Optimalisasi
Tanpa Optimalisasi
Baca Video


Grafik Durasi proses (5)
Durasi waktu (msec)
250
200
150
100
50
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
Data
Optimalisasi
Tanpa Optimalisasi
Baca Video


Grafik durasi proses (6)
Durasi watu (msec)
250
200
150
100
50
0
1
42
83
124
165
206
247
288
329
370
411
452
493
534
575
616
657
698
739
780
821
862
Data
Optimalisasi
Tanpa Optimalisasi
Baca Video

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Rachma aji solicha, lahir di kota Sidoarjo pada tanggal 17 September 1998. Anak pertama dari enam bersaudara, bertempat tinggal di Desa Jumputrejo, RT27/ RW09 Kecamatan Sukodono, Kabupaten Sidoarjo, Jawa Timur. Pernah menempuh pendidikan di MI Tarbiyatus Syarifah, SMP Negeri 1 Buduran, dan SMA Negeri 3 Sidoarjo. Saat ini tengah menempuh jenjang pendidikan di Institut Teknologi Sepuluh Nopember Surabaya, dengan Program Studi Elektro Otomasi, Departemen Teknik Elektro Otomasi, Fakultas Vokasi. Mempunyai hobi membaca. Kegiatan favorit membaca novel dan memasak untuk keluarga. Memiliki cita-cita agar dapat bermanfaat untuk negeri tercinta dan orang-orang sekitar.

E-mail : ajisolicha@gmail.com