TUGAS AKHIR– KM184801

PENERAPAN METODE GUIDED FILTER UNTUK MEREDUKSI NOISE HUJAN PADA VIDEO DIGITAL

IZAH AMALIA
NRP 06111540000068

Dosen Pembimbing :
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

DEPARTEMEN MATEMATIKA
Fakultas Matematika Komputasi dan Sains Data
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT– KM184801

APPLICATION OF GUIDED FILTER METHOD FOR RAIN NOISE REDUCTION BASED ON DIGITAL VIDEO

IZAH AMALIA
NRP 06111540000068

Supervisor :
Dr. Dwi Ratna Sulistyaningrum, S.Si, MT

DEPARTMENT OF MATHEMATICS
Faculty of Mathematics, Computing, and Data Science
Sepuluh Nopember Institute of Technology
Surabaya 2019

LEMBAR PENGESAHAN

PENERAPAN METODE GUIDED FILTER UNTUK MEREDUKSI NOISE HUJAN PADA VIDEO DIGITAL

APPLICATION OF GUIDED FILTER METHOD FOR NOISE RAIN REDUCTION BASED ON DIGITAL VIDEO

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Untuk memperoleh gelar Sarjana Matematika
Pada bidang studi Ilmu Komputer
Program Studi S-1 Departemen Matematika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :
IZAH AMALIA
06111540000068

Menyetujui,

Dosen Pembimbing,

Dr. Dwi Ratna Sulistyaningrum, S.Si. MT
NIP. 19690405 199403 2 003

Mengetahui,
Kepala Departemen Matematika
FMKSD ITS

Dr. Imam Mukhlash, S.Si, MT
NIP. 19700831 199403 1 003
Surabaya, Juli 2019

PENERAPAN METODE GUIDED FILTER UNTUK MEREDUKSI NOISE HUJAN PADA VIDEO DIGITAL

Nama : **Izah Amalia**
NRP : **06111540000068**
Departemen : **Matematika FMKSD - ITS**
Pembimbing : **Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

ABSTRAK

Video adalah salah satu media untuk mengabadikan momen. Namun terkadang ada hambatan pada saat pengambilan video. Salah satu faktor yang menghambat adalah hujan. Butiran-butiran hujan akan menjadi *noise* dan menutupi video yang sesungguhnya sehingga informasi yang didapat pada video tidak tersampaikan secara sempurna, padahal informasi yang didapatkan dari video tersebut dapat dimanfaatkan untuk *tracking object*, identifikasi objek, dan pengenalan objek. *Noise* adalah titik-titik pada citra yang sebenarnya bukan merupakan bagian dari citra, melainkan ikut tercampur pada citra karena suatu sebab. Metode *Guided Filter* sendiri adalah *edge-preserving smoothing filter* yang membuat tepi-tepi objek pada citra lebih halus tanpa *noise*. *Guided Filter* menggunakan *guidance image* atau citra acuan untuk memproses citra input. Pada Tugas Akhir ini, telah dilakukan reduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter*. Metode ini dilakukan sebanyak tiga kali, yaitu pada saat *Pre-Processing*, *Filtering*, dan *Recovering*. Namun hasil terbaik didapatkan pada saat *Filtering*. Untuk mengetahui keakuratan dari metode ini, dilakukan dengan menghitung PSNR antara video hasil dengan video asli tanpa *noise* hujan. PSNR yang didapatkan dalam dinterval 16-26 dB.

Kata Kunci: Reduksi *noise* hujan, *Guided Filter*, *Peak Signal to Noise Ratio* (PSNR).

APPLICATION OF GUIDED FILTER METHOD FOR RAIN NOISE REDUCTION BASED ON DIGITAL VIDEO

Name	**: Izah Amalia**
NRP	**: 06111540000068**
Department	**: Mathematics FMKSD - ITS**
Supervisor	**: Dr. Dwi Ratna Sulistyaningrum, S.Si, MT**

ABSTRACT

Video is one of many media to capture moments. But sometimes there are obstacles when people took videos. One of that obstacles is rain. Rain drops will be noises and cover the real videos so that the information on that videos can not be delivered perfectly. Besides, the information from the videos are so useful for some conditions, such as can be used as object tracking, object identification, and many more. Noises are dots on image which actually are not the part of the image, but got mixed on image because of some reasons. Guided Filter method is an edge-preserving smoothing filter that makes the edges of objects on image become smoother without noise. This method used guidance image to process the input image. In this final project, rain reduction based on digital video has been done using Guided Filter. Guided Filter used three times, in Pre-Processing step, Filtering step, and Recovering step. But the best result got from the Filtering step. PSNR between the result videos and the real videos without the rain noises was used for knowing the accuracy of this method. The result of PSNR are on interval 16-26 dB.

Keywords*: Rain Reduction, Guided Filter, Peak Signal to Noise Ratio (PSNR)*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillaahirobbil'aalamiin, segala puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan limpahan rahmat, taufik dan hidayah – Nya, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul Penerapan Metode Guided Filter untuk Mereduksi *Noise* Hujan pada Video Digital sebagai salah satu syarat kelulusan Program Sarjana Departemen Matematika FMKSD Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Tugas Akhir ini dapat terselesaikan dengan baik dan tepat waktu berkat bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis menyampaikan ucapan terimakasih kepada :

1. Orang tua penulis, Bapak Soedarnanto dan Ibu Hani'ah yang telah mendampingi penulis sejak penulis lahir di dunia hingga melangkah sejauh ini. Tanpa doa dan senyuman beliau setiap pagi, penulis tidak akan bisa berada di titik ini. Juga untuk Zamzam Rahmatulullah, kakak terbaik yang pernah ada, yang selalu menjadi pendengar setia penulis selama ini. Tidak lupa untuk kakak ipar tercinta, Dina Rosita, yang memberi warna baru pada keluarga penulis.
2. Bapak Dr. Imam Mukhlas, S.Si, MT sebagai Kepala Departemen Matematika FKMSD ITS
3. Ibu Dr. Dwi Ratna Sulistyaningrum, S.Si, MT selaku dosen pembimbing yang telah memberikan arahan dan nasehat dengan sangat sabar sehingga penulis bisa menyelesaikan Tugas Akhir ini dengan lancer.
4. Bapak Dr. Darmaji, S.Si, MT, Bapak Drs. Daryono Budi Utomo, MSi, Bapak Drs. Soetrisno, MI.Komp, dan Ibu Prof. DR. Erna Apriliani, M.Si. selaku dosen penguji yang

telah memberi kritik, saran, dan masukan yang membangun dalam menyelesaikan Tugas Akhir.

5. Bapak Dr. Didik Khusnul Arif, S.Si, M.Si selaku Kaprodi S1 Departemen Matematika ITS dan Bapak Drs. Iis Herisman, M.Si selaku Sekprodi S1 Departemen Matematika ITS yang telah bekerja keras untuk menyusun jadwal seminar proposal, seminar hasil, hingga ujian akhir penulis dan teman-teman penulis.
6. Ibu Dra. Nuri Wahyuningsih, M.Kes dan Bapak Drs. Suharmadi, Dipl. Sc, M.Phil selaku dosen wali penulis.
7. Bapak dan Ibu Dosen serta Staff Departemen Matematika ITS.
8. Rachmat Wahyudi Ismail, Mas Eko, dan Mas Adit yang sangat membantu penulis untuk mengerjakan Tugas Akhir, serta sahabat-sahabat penulis Fina Nurul, Isabella, Li'Izza, Nida, Inayah, Bekay, Riko, Danni, Komting, Vira, Syahira, Mbacung, dan Safir yang telah memberi canda tawa di tengah-tengah kehidupan penulis yang *flat*.
9. Luthfiana, 'Alima, Al Farizi, Syifa, Desynta, Aldo, Dhono, Jarmed, Mamnuah, dan Ardi selaku Keluarga Cemara, tempat berbagi suka dan duka tentang citra digital.
10. DOHMAIn tersayang, yang memberi makna apa itu keluarga, teman, dan kerabat, yang memberi cerita indah di kehidupan perkulihaan.
11. Biru si Laptop, Hore si Revo, EXO, dan semua pihak yang telah memberikan dukungan kepada penulis dalam masa perkuliahan hingga penyelesian Tugas Akhir ini.

Penulis menyadari bahwa dalam Tugas Akhir ini masih terdapat kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan. Akhirnya penulis berharap semoga Tugas Akhir ini dapat bermanfaat bagi banyak pihak.

Wassalamu'alaikum Wr. Wb.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini membahas latar belakang secara umum yang mendasari penulisan Tugas Akhir mengenai reduksi *noise* hujan dengan menggunakan metode *Guided Filter* pada video digital. Kemudian dijabarkan juga rumusan masalah, batasan masalah, tujuan, dan manfaat yang diambil berdasarkan latar belakang penyusunan Tugas Akhir ini.

1.1. Latar Belakang

Mengabadikan momen adalah hal yang menarik untuk dilakukan. Orang sering mengabadikan momen dengan menggunakan kamera meskipun setiap momen pasti akan terbadikan di otak manusia. Saat ini penggunaan kamera tidak hanya dilakukan untuk mengambil foto namun juga untuk pengambilan video. Tetapi terkadang ada beberapa hambatan pada saat pengambilan video tersebut

Cuaca merupakan salah satu hambatan pada saat pengambilan video, tidak terkecuali cuaca pada saat hujan. Hujan dapat membuat video yang dihasilkan kurang optimal karena mengandung butiran air hujan atau yang biasa disebut *noise*. *Noise* tersebut akan menutupi video yang sesungguhnya dan membuat video yang dihasilkan mengalami penurunan kualitas. Akibatnya, informasi yang didapat dari video menjadi tidak sempurna. Padahal dari sebuah video bisa didapatkan informasi yang berguna, seperti pada CCTV di jalan raya, video tersebut bisa digunakan untuk *tracking* kendaraan. Pada CCTV di sebuah parkiran, video bisa digunakan untuk mengindentifikasi pelaku jika terjadi kehilangan barang. Oleh karena itu dibutuhkan suatu media untuk mengurangi atau menghilangkan *noise* hujan yang tertangkap kamera pada saat pengambilan video.

Beberapa penelitian sudah pernah dilakukan untuk mengurangi atau menghilangkan *noise* hujan pada foto. Pada tahun 2014, Dias Yusup Wardana melakukan penelitian yang berjudul "Analisis dan Simulasi Penghilang Hujan Pada Citra Digital Menggunakan *Image Decomposition*" yang menggunakan *Bilateral Filter* [1]. Najiya C A dan Sreeram S pada tahun 2015 melakukan penelitian yang berjudul "*Single Image Rain Removal Using Guided Filter*" yaitu penelitian mengenai penghilang hujan pada sebuah foto atau citra dengan menggunakan metode *Guided Filter* [2]. Sedangkan pada tahun 2012, Jing Xu melakukan penelitian yang berjudul "*Removing Rain and Snow in a Single Image using Guided Filter*." Jing Xu juga membandingkan antara penghilang *noise* hujan dengan metode *Guided Filter* dan *Bilateral Filter*. Hasilnya *Guided Filter* lebih detail dalam menghilangkan *noise* hujan dibanding *Bilateral Filter* [3].

Guided Filter adalah *edge-preserving smoothing filter* yang membuat tepi-tepi objek pada citra lebih halus tanpa *noise*. *Guided Filter* mengandalkan *guidance image* atau citra acuan untuk memproses citra input [4]. Tetapi ada beberapa kondisi yang memungkinkan untuk citra input digunakan sebagai citra acuan.

Selain untuk menghilangkan *noise* hujan, *Guided Filter* juga digunakan untuk melakukan penelitian mengenai menghilangkan kabut dan asap pada tahun 2012 oleh Zheqi Lin dan Xuansheng Wang. *Guided Filter* yang sudah dimodifikasi digunakan untuk menghitung perkiraan transmisi *real-time image* dan video penghilang kabut [5].

Berdasarkan penjelasan diatas, pada tugas akhir ini dilakukan penelitian mengenai penghilang *noise* hujan pada video digital dengan menggunakan metode *Guided Filter*.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah disajikan di atas, permasalahan yang akan dibahas dalam penelitian Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana penerapan *Guided Filter* dalam mendeteksi dan mereduksi *noise* hujan pada video?
2. Bagaimana kinerja *Guided Filter* dalam mereduksi *noise* hujan pada video?

1.3. Batasan Masalah

Dalam Tugas Akhir ini, penulis membatasi permasalahan sebagai berikut:

1. *Input* berupa video dengan format mp4 atau avi.
2. Video dengan *frame rate* 24 fps.
3. Kamera pada posisi diam.

1.4. Tujuan

Tujuan dari penelitian Tugas Akhir ini adalah sebagai berikut:

1. Melakukan deteksi dan mereduksi *noise* hujan pada video dengan menggunakan *Guided Filter*.
2. Menganalisa kinerja *Guided Filter* dalam mereduksi *noise* hujan pada video.

1.5. Manfaat

Manfaat yang didapat dari Tugas Akhir ini adalah sebagai berikut :

1. Menambah pengetahuan mengenai pengolahan citra digital.

2. Membantu penelitian untuk *tracking object* atau pemantauan CCTV pada saat hujan

1.6. Sistematika Penulisan

Penulisan Tugas Akhir ini disusun dalam lima bab, yaitu :

1. BAB I : PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat penelitian reduksi *noise* hujan pada video digital dan sistematika penulisan Tugas Akhir.

2. BAB II : TINJAUAN PUSTAKA

Bab ini menjelaskan teori dasar yang mendukung penulisan Tugas Akhir, yaitu penelitian terdahulu, citra digital, *noise*, video, *Guided Filter*, *Weighted Summation*, *Enhancement*, dan *Peak Signal to Noise Ratio* (PSNR).

3. BAB III : METODE PENELITIAN

Bab ini menjelaskan tentang tahapan-tahapan dan metode yang digunakan, seperti *Guided Filter, Weighted Summation,* dan *Enhancement*, disertai penjelasan dalam tiap tahapan yang dilakukan dalam menyelesaikan Tugas Akhir.

4. BAB IV : PERANCANGAN DAN IMPLEMENTASI PROGRAM

Bab ini membahas tentang perancangan program, yaitu penjelasan kegiatan yang dilakukan sesuai dengan yang dijabarkan pada metode penelitian, seperti bagaimana penggunaan *Guided Filter, Weighted Summation,* dan *Enhancement* pada Tugas Akhir ini, dan implementasi program tersebut ke dalam bahasa komputer.

5. BAB V : UJI COBA DAN ANALISA HASIL

Bab ini menjabarkan data yang digunakan untuk uji coba dan hasil serta Analisa reduksi *noise* hujan pada video digital yang dilakukan pada Tugas Akhir

6. BAB VI : SIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan akhir yang diperoleh dari uji coba dan analisa hasil serta saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini diuraikan mengenai penelitian terdahulu terkait reduksi *noise* hujan penggunaan metode *Guided Filter*, serta teori tentang citra digital, *noise*, video, *Guided Filter*, *Weighted Summation*, *Enhancement*, dan *Peak Signal to Noise Ratio* (PSNR).

2.1. Penelitian Terdahulu

Penelitian mengenai penghilang hujan atau salju pada sebuah foto atau citra bukanlah penelitian yang baru saja dilakukan. Beberapa peneliti terdahulu telah melakukan penelitian seperti ini.

Najiya C A dan Sreeram S melakukan penelitian yang berjudul *Single Image Rain Removal Using Guided Filter* pada tahun 2015,. *Guided Filter* digunakan tiga kali pada penelitian ini, yaitu untuk input citra *low frequency*, citra *high frequency*, dan untuk menyempurnakan hasil akhir citra tanpa *noise* hujan [2]. Untuk *edge enhancement*, digunakan *sobel filter*. Hasil dari penelitian ini adalah citra tanpa *noise* hujan. *Guided Filter* terkadang berdampak negatif, tapi dapat diselesaikan dengan metode *recovery*.

Dias Yusup Wardana melakukan penelitian yang berjudul Analisis dan Simulasi Penghilang Hujan Pada Citra Digital Menggunakan *Image Decomposition* tahun 2014. Pada *Image Decomposition* menggunakan *Bilateral Filter* untuk *smoothing* citra, *Histogram of Oriented Gradient* (HOG) untuk menampilkan karakteristik dstribusi gradien pada citra tersebut, dan algoitma *K-Means* untuk memisahkan komponen hujan dan non hujan [1].

Siti Khotijah melakukan penelitian yang berjudul Penerapan Transformasi Wavelet Daubechies untuk Reduksi

Noise Hujan pada Video. Penelitian ini menggunakan metode Wavelet Daubechies yang terdiri dari beberapa proses, yaitu Dekomposisi Wavelet, proses Fusi, proses Thresholding dan proses Rekonstrusi [6].

Jing Xu, Wei Zhao, Peng Liu, dan Xiangdong Tang melakukan penelitian yang berjudul *Removing Rain and Snow in a Single Image using Guided Filter* pada tahun 2012. Hasil dari penelitian ini dibandingkan dengan penelitian mereka sebelumnya, yaitu menghilangkan hujan atau salju dengan *Image Decomposition* yang menggunakan *Bilateral Filter*. Setelah dibandingkan, *Guided Filter* lebih detail dalam penghilangan noise hujan atau salju [3].

Guided Filter tidak hanya digunakan untuk penghilang noise hujan dan salju. Pada tahun 2012, Zheqi Lin dan Xuansheng Wang melakukan penelitian yang berjudul *Dehazing for Image and Video Using Guided Filter*. Penelitian ini bertujuan untuk menghilangkan kabut dan asap pada sebuah citra atau video dengan membandingkan bagian-bagian yang gelap pada citra tersebut [5]. Bagian-bagian yang lebih gelap dianggap bagian yang tertutup kabut atau asap dan selanjutnya kabut atau asap tersebut akan dihilangkan dengan *Guided Filter*.

2.2. Landasan Teori

Sebelum membahas lebih dalam mengenai reduksi *noise* hujan pada video digital, perlu diketahu beberapa teori dasar yang berhubungan dengan penelitian ini, seperti teori tentang citra digital, *noise*, video, *guided filter*, *weighted summation*, *enhancement*, dan PSNR. Berikut merupakan penjabaran dari teori-teori tersebut.

2.2.1. Video

Video adalah teknologi untuk menangkap, merekam, memproses, mentransmisikan dan menata ulang gambar bergerak yang biasanya menggunakan film, seluloid, sinyal elektronik, atau media digital [6]. Ada dua macam video, yaitu:

1. Video Analog

   Video analog tersusun dari gelombang bersambung yang bervariasi, dengan kata lain nilai sinyal akan memiliki angka yang beragam tetapi terbatas pada batas maksimum dan minimum yang diijinkan [7].

2. Video Digital

   Video digital ditransmisikan hanya berupa titik presisi yang dipilih pada interval dalam kurva. Tipe sinyal digital yang dapat dipakai oleh komputer kita adalah tipe *binary* [7].

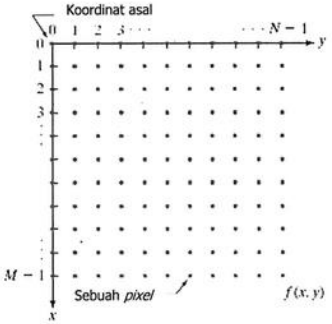
Pada penelitian ini video yang digunakan adalah video digital. Video digital pada dasarnya tersusun atas serangkaian citra digital diam (*frame*) berurutan yang direkam dan ditampilkan secara bersambung dalam suatu satuan waktu dengan kecepatan tertentu. Jika laju *frame* tinggi maka dapat dilihat sebagai rangkaian gerak yang kontinu sehingga pergerakan obyek yang ada dalam citra digital tersebut seolah-olah bergerak. Kecepatan dari *frame* disebut *frame rate* dengan satuan fps (*frame per second*). Semakin besar nilai *frame rate* maka akan semakin halus pergerakan yang ditampilkan.

2.2.2. Pengertian Citra Digital

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan computer. Dalam konteks yang lebih luas, pengolahan citra digital

mengacu pada pemrosesan setiap data dua dimensi. Citra digital merupakan larik (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu [8].

Suatu citra dapat didefinisikan sebagai fungsi *f(x,y)* berukuran M baris dan N kolom, dengan x dan y sebagai koordinat spasial, dan amplitude *f* di titik koordinat *(x,y)* dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitude f secara keseluruhan berhingga *(finite)* dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Piksel pada citra digital dapat dilihat pada Gambar 2.1.



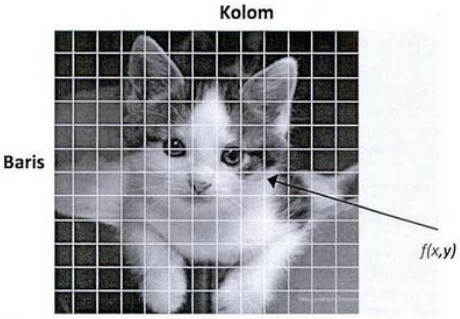
Gambar 2.1 Piksel pada sebuah citra [8]

Citra digital dapat dituls dalam bentuk matrik sebagai berikut

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \ldots.$$

Nilai pada suatu irisan antara baris dan kolom (pada posisi x,y) disebut dengan *picture elements, image elements, pels,* atau

pixels. Istilah terakhir (pixel) paling sering digunakan pada citra digital. Ilustrasi citra digital dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi digitalisasi citra [8]

2.2.3. *Noise*

Noise adalah titik-titik pada citra yang sebenarnya bukan merupakan bagian dari citra, melainkan ikut tercampur pada citra karena suatu sebab [9]. Ada tiga macam *noise*, yaitu:

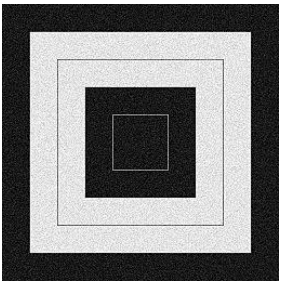
a. *Noise Aditif*

Noise aditif adalah *noise* yang ditambahkan secara seragam atau menambahkan *noise* secara merata pada sebuah bidang citra dengan varian tertentu. Contoh *noise* ini adalah *noise salt-and-peppers* yang menambahkan aras gelap dan terang pada citra.

Gambar 2.3 Contoh salt-and-pepper noise

b. *Noise Gaussian*

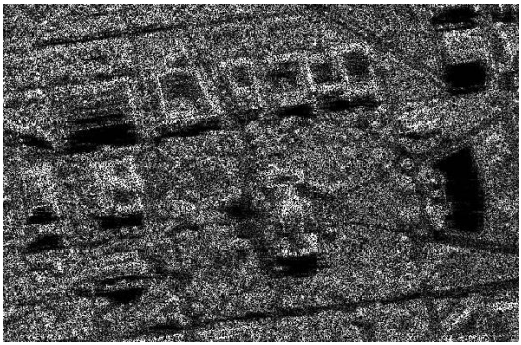
Noise ini memiliki intensitas yang sesuai dengan distribusi normal yang memiliki rerata (*mean*) dan varian tertentu.



Gambar 2.4 Contoh gaussian noise

c. *Noise Speckle*

Noise ini muncul pada saat pengambilan citra tidak sempurna karena alasan cuaca, perangkat pengambil citra dan sebagainya. Sifat *noise* ini *multiplikatif*, artinya semakin besar intensitas citra atau semakin cerah citra, semakin jelas juga *noise* yang terlihat.

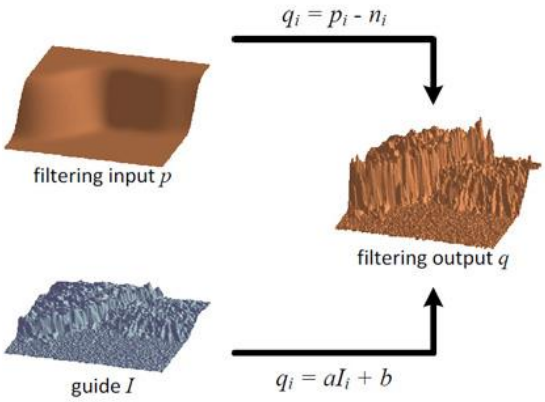


Gambar 2.5 Contoh speckle noise

Noise muncul biasanya sebagai akibat dari gangguan yang tidak sesuai (sensor *noise*, *photographic gain noise*). Gangguan tersebut umumnya berupa variasi intensitas suatu piksel yang tidak berkorelasi dengan piksel-piksel tetangganya. Secara visual, gangguan mudah dilihat oleh mata karena tampak berbeda dengan piksel tetangganya. Piksel yang mengalami gangguan umumnya memiliki frekuensi tinggi. Komponen citra yang berfrekuensi rendah umumnya mempunyai nilai piksel konstan atau berubah sangat lambat. Operasi *denoise* dilakukan untuk menekan komponen yang berfrekuensi tinggi dan meloloskan komponen yang berfrekuensi rendah.

2.2.4. *Guided Filter*

Guided Filter adalah algoritma *non-approximate linear-time* yang menghasilkan *filtering output* dengan mempertimbangkan konten pada *guidance image* atau citra acuan, yang bisa menjadi input citra itu sendiri [4]. *Guided Filter* adalah *low pass filter* yang berarti dapat menghasilkan citra *low frequency*.



Gambar 2.6 Ilustrasi guided filter [10]

Ilustrasi *Guided Filter* ditunjukkan pada Gambar 2.6. *Guided Filter* merupakan model lokal linear antara citra acuan I dan *output* q. Model tersebut ditujukan pada persamaan (2.1).

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \quad (2.1)$$

dengan q_i adalah *output* pada piksel i, I_i adalah citra acuan pada piksel i, ω_k adalah *window* pada pusat piksel k, serta a_k dan b_k adalah koefisien linear yang memiliki nilai konstan pada ω_k. *Window* yang digunakan adalah *window* persegi dengan radius r.

Dalam menentukan koefisien linear a_k dan b_k pada persamaan (2.1), diperlukan batasan dari *output* q. *Ouput* q dimodelkan sebagai *input* p untuk mengurangi komponen n yang tidak diinginkan. Model tersebut dapat ditulis pada persamaan (2.2)

$$q_i = p_i - n_i \quad (2.2)$$

dengan q_i adalah *output* pada piksel i, p_i adalah *input* pada piksel i, dan n adalah *noise* pada piksel i yang tidak diinginkan.

Untuk menentukan koefisien linear a_k dan b_k, dicari solusi untuk meminimalisir n_i atau selisih antara *input* p dan *output* q. Sehingga didapatkan persamaan model linear ridge regression [10] yang ditunjukkan pada persamaan (2.3)

$$E(a_k, b_k) = \sum_{i \in \omega_k} \left((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2 \right) \qquad (2.3)$$

dengan ϵ merupakan parameter regularisasi atau parameter untuk mencegah a_k terlalu besar.

Dari persamaan (2.3), didapatkan solusi (2.4) dan (2.5).

$$a_k = \frac{\sum_{i \in \omega_k} \frac{I_i p_i - \mu_k \bar{p}_k}{|\omega_k|}}{\sigma_k^2 + \epsilon} \qquad (2.4)$$

$$b_k = \bar{p}_k - a_k \mu_k \qquad (2.5)$$

dengan μ_k dan σ_k^2 adalah mean dan varian I di ω_k, $|\omega_k|$ adalah jumlah piksel ω_k dan $\bar{p}_k = \sum_{i \in \omega_k} \frac{p_i}{|\omega_k|}$ adalah mean p di ω_k.

Setelah menghitung (a_k, b_k) untuk semua ω_k pada citra, didapatkan *filter output* seperti pada persamaan (2.6).

$$\hat{q}_i = \sum_{k: i \in \omega_k} \frac{(a_k I_i + b_k)}{|\omega_k|} = \bar{a}_i I_i + \bar{b}_i \qquad (2.6)$$

Guided Filter adalah *edge-preserving smoothing filter* seperti *Bilateral Filter* tetapi *Guided Filter* memiliki O(N) *time algorithm* yang memiliki kulitas sangat baik dan efisien untuk beberapa aplikasi [2]. Contoh aplikasi yang menggunakan *Guided Filter* adalah *image denoising*, *image enhancement*, *image feathering*, dan *haze removal*. Metode ini hanya bisa digunakan untuk citra *grayscale*, sehingga untuk memproses citra RGB, masing-masing komponen R, G, dan B akan diproses satu satu dengan *Guided Filter*.

2.2.5. *Weighted Summation*

Weighted summation atau penjumlahan berbobot adalah *compensatory method* atau metode pengganti, yang berarti kriteria yang dianggap memiliki nilai buruk dapat diganti dengan yang memiliki nilai yang baik [11]. *Weighted summation* pada citra digital biasanya digunakan untuk menggabungkan dua citra untuk mengambil bagian-bagian terbaik dari kedua citra tersebut. Pada penelitian yang dilakukan oleh Xianhui Zheng, persamaan *weighted summation* yang digunakan ditunjukkan pada persamaan (2.7).

$$I_{ws} = \beta I_a + (1 - \beta) I_b \tag{2.7}$$

dengan I_{ws} adalah hasil dari *weighted summation*, I_a dan I_b adalah citra yang akan digabungkan, dan β adalah bobot yang diinginkan [4].

2.2.6. *Enhancement*

Enhancement adalah proses manipulasi pada gambar agar hasilnya lebih cocok digunakan untuk aplikasi spesifik dibanding dengan gambar asli [12]. Metode *enhancement* yang bisa digunakan bervariasi sehingga untuk mengaplikasikan metode ini harus disesuaikan dengan masalah yang dihadapi. *Image Enhancement* atau peningkatan kualitas citra dapat dilakukan dengan berbagai macam metode [13]. Salah satu dari berbagai macam metode tersebut adalah *Edge Crispening*. *Edge Crispening* digunakan untuk menonjolkan tepi-tepi objek yang terdapat pada citra agar tepi-tepi tersebut bisa diatur tingkat kecerahannya. Selain itu, terdapat metode *Noise Cleaning* yang digunakan untuk menghilangkan *noise* yang terkandung dalam suatu citra agar citra tersebut lebih jelas tanpa adanya *noise*.

Tidak ada teori pasti untuk menggunakan *enhancement*. Ketika suatu gambar diproses, kita sendiri yang menentukan metode *enhancement* seperti apa yang cocok. Contohnya, metode yang digunakan untuk *enhancing* gambar X-ray dengan

baik belum tentu dapat digunakan dengan baik juga untuk *enhancing* gambar satelit yang didapat melalui pita infamerah pada spektrum elektromagnetik. Namun, ketika berhungan dengan *machine perception*, metode *enhancement* lebih mudah dilakukan.

2.2.7. *Peak Signal to Noise Ratio* (PSNR)

Peak Signal to Noise Ratio atau PSNR adalah suatu metode untuk menghitung perbandingan antara nilai maksimum kedalaman bit citra yang diukur dengan besarnya *noise* yang berpengaruh pada citra tersebut [14]. PSNR biasanya diukur dalam satuan desibel (dB). PSNR digunakan untuk mengetahui perbandingan kualitas citra tanpa *noise* dan citra sesudah diproses. Semakin besar nilai PSNR, hasil pemrosesan citra semakin bagus dan mendekati citra aslinya. Perhitungan nilai PSNR ditunjukkan pada persamaan (2.8).

$$PSNR = 20 \log_{10} \left(\frac{255}{\sqrt{MSE}} \right) \tag{2.8}$$

Besarnya *noise* tersebut diwakili dengan nilai *Mean Square Error* atau MSE. MSE adalah ukuran yang digunakan untuk menilai baik tidaknya sebuah metode untuk melakukan rekontruksi atau restorasi citra relative terhadap citra aslinya. Semakin kecil nilai MSE, hasil pemrosesan citra semakin bagus. Perhitungan nilai MSE dapat dilakukan sesuai persamaan (2.9).

$$MSE = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} [f_1(x,y) - f_2(x,y)]^2 \tag{2.9}$$

dengan M dan N merupakan jumlah baris dan kolom pada citra *input* serta $f_1(x,y)$ dan $f_2(x,y)$ adalah citra asli dan citra *output* yang sudah diproses.

Penelitian Subham R. Goratarkar dan Aarti S. Gaikwad mengenai pemrosesan video dengan menggunakan metode *Guided Filter* menghasilkan PSNR sebesar 17-19 dB [15]. Metode ini lebih baik dibandingkan dengan metode Bilateral Filter yang juga mempunyai PSNR sebesar 17-19 dB namun lebih kecil nilai komanya.

BAB III
METODE PENELITIAN

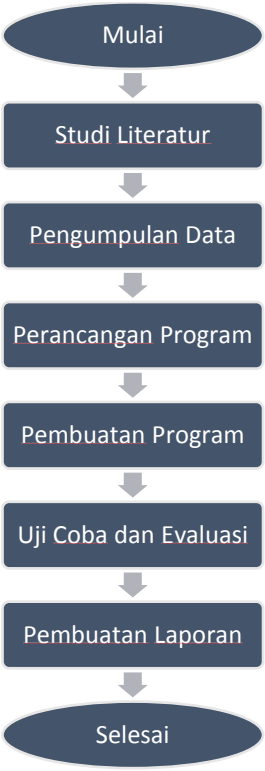
Bab ini menguraikan objek dan data yang digunakan dalam Tugas Akhir reduksi *noise* hujan dengan menggunakan metode *Guided Filter* pada video digital serta langkah-langkah sistematis yang dilakukan dalam proses pengerjaan Tugas Akhir ini.

3.1. Data Penelitian

Objek penelitian dalam Tugas Akhir ini adalah video dengan *noise* hujan yang didapat dari pengambilan data video sendiri. *Noise* hujan yang digunakan adalah *noise* hujan buatan. Penambahan *noise* hujan dilakukan dengan cara menambahkan video *rain overlay* ke dalam video asli dengan menggunakan program Adobe Premier Pro.

3.2. Tahapan Penelitian

Kegiatan penelitian dalam Tugas Akhir ini melalui tahapan-tahapan studi literatur, pengumpulan data, perancangan program, pembuatan program, uji coba dan evaluasi, dan pembuatan laporan, seperti yang ditunjukkan pada Gambar 3.1



Gambar 3.1 Langkah-langkah penelitian

Berikut merupakan penjelasan lebih lanjut mengenai langkah-langkah penelitian reduksi *noise* hujan pada video digital :

a. Studi Literatur

Sebelum membuat program untuk mereduksi noise hujan, akan dilakukan studi literatur terlebih dahulu untuk mengetahui lebih jauh apa saja yang dilakukan untuk membuat program tersebut. Studi literatur yang dilakukan meliputi *Guided Filter*, *Edge-enhancement,* dan *weight*

summation. Selama melakukan penelitian pun juga akan terus dilakukan studi literatur untuk mendapatkan hasil penelitian yang lebih maksimal.

b. Pengumpulan Data

Data adalah hal yang sangat krusial dalam pembuatan penelitian, begitu juga dengan penelitian mengenai reduksi noise hujan ini. Seiring dengan melakukan studi literatur, akan dilakukan pengumpulan data sebagai dasar untuk pembuatan program. Data yang dikumpulkan adalah video yang mengandung noise hujan yang didapat dari video yang direkam sendiri oleh penulis dan data video dari *github*.

c. Perancangan Program

Suatu program tidak dibuat secara langsung. Dibutuhkan rancangan program reduksi noise hujan agar program yang dibuat dapat digunakan dengan baik.

d. Pembuatan Program

Setelah mengumpulkan data dan merancang program, akan mulai dilakukan pembuatan program. Langkah-langkah algoritma reduksi *noise* hujan pada video digital ditunjukkan pada Gambar 3.2.

Gambar 3.2 Diagram Alir Pereduksi *Noise* Hujan

Penjelasan lebih lanjut mengenai langkah-langkah tersebut adalah sebagai berikut :

1. Input Video dengan *Noise* Hujan

   Pada program ini input berupa video hujan yang pada akhirnya *noise* hujan tersebut akan dihilangkan.

2. Memproses Video menjadi *Frame*

   Sebelum melakukan proses menghilangkan *noise* hujan, video tersebut akan dirubah menjadi frame-frame.

3. *Guided Filter* Pertama

   Pada proses *Guided Filter* pertama ini, frame-frame input menjadi citra acuan. Karena *Guided Filter* merupakan *low pass filter*, maka otomatis akan didapat citra *low frequency*. Lalu citra acuan akan dikurangkan dengan citra *low frequency* untuk mendapat citra *high frequency*. Selanjutnya, tingkat *smoothness* dari citra *low frequency* akan diseting untuk mendapatkan citra *low frequency* tanpa *noise* hujan.

4. Menajamkan Tepi dengan *Edge Enhancement*

   Hasil dari *Guided Filter* yang pertama ini akan membuat tepi-tepi dari citra *low frequency* menjadi sangat halus dan cenderung sedikit kabur. Sehingga akan dilakukan *edge enhancement* untuk membuat tepi-tepi tersebut lebih terlihat. Citra yang telah diproses *edge enhancement* ini akan dijadikan citra acuan pada *Guided Filter* selanjutnya.

5. *Guided Filter* Kedua

Pada proses *Guided Filter* yang kedua, citra *high frequency* akan menjadi citra input. Dengan citra hasil *edge enhancement* sebagai citra acuan, didapat citra *high frequency* tanpa *noise* hujan.

6. Menyetarakan setiap bagian citra dengan *Weight Summation*

Setelah proses *Guided Filter* yang kedua, bagian-bagian *noise* hujan menjadi lebih terang dari bagian lainnya. Untuk mengatasi masalah ini, akan dilakukan *weight summation* pada citra hasil *Guided Filter* yang kedua tersebut. Hasil dari *weight summation* ini akan menjadi citra acuan untuk *Guided Filter* terakhir.

7. *Guided Filter* Ketiga

Guided Filter terakhir ini dilakukan untuk mendapat hasil maksimal. Citra input berupa citra hasil *Guided Filter* kedua dan citra acuan adalah citra hasil *weight summation*. Hasil dari *Guided Filter* terakhir ini adalah citra tanpa *noise* hujan.

8. Memproses Video menjadi Frame

Frame-frame yang telah diproses penghilangan *noise* hujan selanjutnya akan dijadikan video lagi.

9. *Output* Video reduksi *Noise* Hujan

Video yang dihasilkan dari penyatuan frame-frame tersebut adalah video tanpa *noise* hujan.

10. Hitung Nilai PSNR

Setelah didapatkan *output* video dengan *noise* hujan yang sudah tereduksi, video tersebut

dibandingkan dengan video asli tanpa noise hujan dan dihitung nilai MSE untuk mendapatkan nilai PSNR.

e. Uji Coba dan Evaluasi

Setelah program reduksi noise hujan ini selesai dibuat, akan dilakukan uji coba program apakah program ini sudah valid atau belum dan apakah pengguna dapat menggunakan program ini dengan baik atau belum. Disamping uji coba, akan dilakukan juga evaluasi pada program agar program lebih maksimal.

f. Pembuatan Laporan

Di akhir penelitian ini, akan dilakukan pembuatan laporan. Laporan dari program penghilang noise hujan ini akan dibuat setelah program lolos dari uji coba dan telah menjawab evaluasi yang didapat.

BAB IV
PERANCANGAN DAN IMPLEMENTASI PROGRAM

Bab ini menjelaskan lebih lanjut mengenai perancangan dan implementasi program reduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter*.

4.1. Perancangan Program

Langkah-langkah yang dilakukan untuk membuat reduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter* adalah dengan mengaplikasikan metode *Guided Filter*, *Weighted Summation*, dan *Edge Enhancement*. Sub-bab ini akan membahas secara detail bagaimana langkah-langkah tersebut dilakukan.

Gambar 4.1 merupakan contoh citra Cameraman yang didapat dari penyimpanan di Matlab. Pada pemahasan perancangan program ini, akan diambil citra *input* berupa matriks 6x6 dari citra Cameraman terebut, seperti pada Gambar 4.2.

Gambar 4.1 Cameraman

Gambar 4.2 Citra input 6x6

Potongan dari gambar tersebut mempunyai matriks sebagai berikut :

$$I_n = \begin{bmatrix} 0.5608 & 0.5608 & 0.5255 & 0.5412 & 0.6196 & 0.3294 \\ 0.5412 & 0.5412 & 0.5137 & 0.5176 & 0.5843 & 0.5647 \\ 0.5137 & 0.5137 & 0.5216 & 0.5451 & 0.5922 & 0.6471 \\ 0.4902 & 0.5176 & 0.5059 & 0.5137 & 0.6078 & 0.6784 \\ 0.5294 & 0.5490 & 0.5373 & 0.4235 & 0.4784 & 0.6078 \\ 0.6627 & 0.6627 & 0.6431 & 0.4745 & 0.3451 & 0.4902 \end{bmatrix}$$

Matriks tersebut diproses sesuai dengan diagram alir yang ditunjukan pada Gambar 3.2. Berikut penjelasan lebih lanjut mengenai proses tersebut :

a. *Guided Filter* pertama

Matriks input tersebut diproses Guided Filter yang pertama dengan citra input menjadi citra acuan juga. Nilai a_{k_1} dan b_{k_1} yang dihitung sesuai persamaan (2.4) dan (2.5) didapatkan :

$$a_{k_1} = \begin{bmatrix} 0.0320 & 0.0278 & 0.0880 & 0.3191 & 0.3656 & 0.4276 \\ 0.0408 & 0.0357 & 0.1089 & 0.3231 & 0.3678 & 0.4196 \\ 0.0367 & 0.0780 & 0.1411 & 0.3215 & 0.3684 & 0.4294 \\ 0.2353 & 0.2489 & 0.3178 & 0.3547 & 0.3775 & 0.4195 \\ 0.2706 & 0.2899 & 0.3611 & 0.4007 & 0.4234 & 0.4664 \\ 0.3013 & 0.3464 & 0.4194 & 0.4410 & 0.4557 & 0.4850 \end{bmatrix}$$

$$b_k = \begin{bmatrix} 0.5154 & 0.5182 & 0.4981 & 0.3685 & 0.3437 & 0.3143 \\ 0.5040 & 0.5077 & 0.4824 & 0.3703 & 0.3480 & 0.3261 \\ 0.5087 & 0.4823 & 0.4585 & 0.3674 & 0.3428 & 0.3139 \\ 0.4202 & 0.4025 & 0.3636 & 0.3505 & 0.3359 & 0.3123 \\ 0.4041 & 0.3818 & 0.3395 & 0.3253 & 0.3104 & 0.2847 \\ 0.3958 & 0.3546 & 0.3074 & 0.2994 & 0.2860 & 0.2644 \end{bmatrix}$$

Nilai a_{k_1} dan b_{k_1} tersebut digunakan untuk mendapatkan filter *output* q_1 sesuai persamaan (2.6) sebagai hasil dari *Guided Filter* yang pertama.

$$q_1 = \begin{bmatrix} 0.5340 & 0.5377 & 0.5341 & 0.5407 & 0.5678 & 0.4662 \\ 0.5345 & 0.5365 & 0.5320 & 0.5337 & 0.5566 & 0.5526 \\ 0.5317 & 0.5320 & 0.5338 & 0.5411 & 0.5592 & 0.5838 \\ 0.5265 & 0.5326 & 0.5289 & 0.5293 & 0.5644 & 0.5963 \\ 0.5360 & 0.5417 & 0.5381 & 0.4955 & 0.5137 & 0.5665 \\ 0.5799 & 0.5823 & 0.5766 & 0.5116 & 0.4565 & 0.5159 \end{bmatrix}$$

q atau hasil *Guided Filter* yang pertama ini juga bisa disebut sebagai citra *low frequency*, mengingat *Guided Filter* memiliki sifat *low pass filter*. Dari citra *low frequency* ini bias didapatkan citra *high frequency* dengan cara mengurangkan citra input dengan citra *low frequency*. Matriks dari citra *high frequency* yang didapatkan adalah :

$$I_{HF} = \begin{bmatrix} 0.0268 & 0.0231 & 0 & 0.0005 & 0.0518 & 0 \\ 0.0066 & 0.0046 & 0 & 0 & 0.0277 & 0.0121 \\ 0 & 0 & 0 & 0.0040 & 0.0330 & 0.0633 \\ 0 & 0 & 0 & 0 & 0.0434 & 0.0821 \\ 0 & 0.0073 & 0 & 0 & 0 & 0.0414 \\ 0.0829 & 0.0805 & 0.0665 & 0 & 0 & 0 \end{bmatrix}$$

b. *Edge Enhancement*

Edge Enhancement digunakan untuk mendeteksi tepi-tepi objek yang terdapat pada citra. Metode *edge enhancement* yang digunakan adalah *sobel filter*. Citra hasil dari *Guided Filter* yang pertama atau citra *low frequency* diproses dengan *sobel filter*, didapatkan matriks:

$$\nabla I_{LF} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

dengan ∇I_{LF} adalah hasil dari *sobel* filter.

Hasil dari *sobel filter* akan dikembalikan menjadi citra grayscale dengan rumus :

$$I_{LF}^{*} = I_{LF} + \omega \nabla I_{LF}$$

dengan I_{LF} adalah citra *low frequency* dan $\omega = 0.1$.

$$I_{LF}^{*} = \begin{bmatrix} 0.5340 & 0.5377 & 0.5341 & 0.5407 & 0.5678 & 0.4662 \\ 0.5345 & 0.5365 & 0.5320 & 0.5337 & 0.5566 & 0.5526 \\ 0.5317 & 0.5320 & 0.5338 & 0.5411 & 0.5592 & 0.5838 \\ 0.5265 & 0.5326 & 0.5289 & 0.5293 & 0.5644 & 0.5963 \\ 0.5360 & 0.5417 & 0.5381 & 0.4955 & 0.5137 & 0.5665 \\ 0.5799 & 0.5823 & 0.5766 & 0.5116 & 0.4565 & 0.5159 \end{bmatrix}$$

c. *Guided Filter* kedua

Pada *Guided Filter* kedua citra input yang digunakan adalah citra *high frequency* dan citra acuannya adalah citra hasil dari *edge enhancement* atau I_{LF}^{*}. Dengan cara yang sama pada *Guided Filter* pertama, didapatkan nilai a_{k_2} dan b_{k_2} serta filter *output* q_2 sebagai berikut :

$$a_{k_2} = \begin{bmatrix} 0.0024 & 0.0023 & 0.0192 & 0.1005 & 0.1234 & 0.1566 \\ 0.0044 & 0.0044 & 0.0249 & 0.1031 & 0.1250 & 0.1526 \\ 0.0044 & 0.0162 & 0.0351 & 0.1029 & 0.1258 & 0.1588 \\ 0.0658 & 0.0718 & 0.1016 & 0.1203 & 0.1323 & 0.1554 \\ 0.0790 & 0.0881 & 0.1219 & 0.1441 & 0.1574 & 0.1840 \\ 0.0913 & 0.1130 & 0.1528 & 0.1672 & 0.1767 & 0.1962 \end{bmatrix}$$

$$b_{k_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$q_2 = \begin{bmatrix} 0 & 0 & 0 & 0.0012 & 0.0518 & 0 \\ 0 & 0 & 0 & 0.0002 & 0.0040 & 0.0044 \\ 0 & 0 & 0 & 0.0006 & 0.0039 & 0.0082 \\ 0 & 0 & 0 & 0 & 0.0033 & 0.0090 \\ 0 & 0.0007 & 0 & 0 & 0 & 0.0034 \\ 0.0063 & 0.0068 & 0.0057 & 0 & 0 & 0 \end{bmatrix}$$

Guided Filter kedua ini menghasilkan citra yang hanya berisi tepi-tepi objek yang telah halus, sehingga ditambahkan dengan citra *low frequency* untuk memulihkan citra tersebut dengan rumus sebagai berikut :

$$I_r = q_1 + q_2$$

Dari rumus tersebut didapatkan :

$$I_r = \begin{bmatrix} 0.5308 & 0.5357 & 0.5326 & 0.5419 & 0.5736 & 0.4601 \\ 0.5322 & 0.5352 & 0.5309 & 0.5339 & 0.5606 & 0.5570 \\ 0.5299 & 0.5310 & 0.5331 & 0.5417 & 0.5631 & 0.5920 \\ 0.5250 & 0.5320 & 0.5279 & 0.5275 & 0.5677 & 0.6053 \\ 0.5360 & 0.5425 & 0.5381 & 0.4884 & 0.5088 & 0.5698 \\ 0.5862 & 0.5891 & 0.5824 & 0.5060 & 0.4418 & 0.5105 \end{bmatrix}$$

Namun, hasil dari pemulihan ini membuat citra sedikit kabur. Sehingga dicari citra yang lebih jelas dengan menggunakan rumus sebagai berikut :

$$I_{clr} = \min(I_r, I_n)$$

dengan I_r merupakan citra yang telah dipulihkan dan I_n merupakan citra input. Hasil akhir dari proses *Guided Filter* kedua ini adalah matriks sebagai berikut :

$$I_{clr} = \begin{bmatrix} 0.5308 & 0.5357 & 0.5255 & 0.5412 & 0.5736 & 0.3294 \\ 0.5322 & 0.5352 & 0.5137 & 0.5176 & 0.5606 & 0.5570 \\ 0.5137 & 0.5137 & 0.5216 & 0.5417 & 0.5631 & 0.5920 \\ 0.4902 & 0.5176 & 0.5059 & 0.5137 & 0.5677 & 0.6053 \\ 0.5294 & 0.5425 & 0.5373 & 0.4235 & 0.4784 & 0.5698 \\ 0.5862 & 0.5891 & 0.5824 & 0.4745 & 0.3451 & 0.4902 \end{bmatrix}$$

d. *Weighted Summation*

Dampak dari hasil *Guided Filter* kedua membuat bagian *noise* hujan menjadi lebih terang dari bagian lainnya sehingga akan dilakukan penyetaraan dengan *Weighted Summation*. Rumus *Weighted Summation* yang digunakan sesuai pada persamaan (2.7).

$$I_{ref} = \beta I_{clr} + (1 - \beta) I_r$$

dengan I_{clr} dan I_r adalah kedua citra yang diproses, I_{ref} adalah hasil *Weighted Summation*, dan $\beta = 0.8$. Matriks yang didapatkan dari *Weighted Summation* ini adalah sebagai berikut :

$$I_{ref} = \begin{bmatrix} 0.5308 & 0.5357 & 0.5269 & 0.5413 & 0.5736 & 0.3556 \\ 0.5322 & 0.5352 & 0.5172 & 0.5209 & 0.5606 & 0.5570 \\ 0.5170 & 0.5172 & 0.5239 & 0.5417 & 0.5631 & 0.5920 \\ 0.4972 & 0.5205 & 0.5103 & 0.5165 & 0.5677 & 0.6053 \\ 0.5307 & 0.5425 & 0.5374 & 0.4365 & 0.4845 & 0.5698 \\ 0.5862 & 0.5891 & 0.5824 & 0.4808 & 0.3644 & 0.4943 \end{bmatrix}$$

e. *Guided Filter* ketiga

Selanjutnya, dilakukan *Guided Filter* yang ketiga untuk mendapatkan hasil yang maksimal. Dengan cara yang sama dengan *Guided Filter* pertama dan kedua, didapatkan nilai a_{k_3} dan b_{k_3} sebagai berikut :

$$a_{k_3} = \begin{bmatrix} 0.0064 & 0.0089 & 0.0302 & 0.2343 & 0.2770 & 0.3396 \\ 0.0137 & 0.0153 & 0.0412 & 0.2112 & 0.2500 & 0.3031 \\ 0.0154 & 0.0551 & 0.0773 & 0.2142 & 0.2538 & 0.3117 \\ 0.0735 & 0.1120 & 0.1967 & 0.2232 & 0.2525 & 0.2989 \\ 0.0873 & 0.1351 & 0.2290 & 0.2586 & 0.2895 & 0.3371 \\ 0.1001 & 0.1701 & 0.2749 & 0.2955 & 0.3205 & 0.3555 \end{bmatrix}$$

$$b_{k_3} = \begin{bmatrix} 0.5213 & 0.5222 & 0.5185 & 0.4038 & 0.3810 & 0.3494 \\ 0.5125 & 0.5139 & 0.5088 & 0.4188 & 0.3992 & 0.3745 \\ 0.5149 & 0.4904 & 0.4843 & 0.4138 & 0.3924 & 0.3630 \\ 0.4947 & 0.4651 & 0.4170 & 0.4083 & 0.3902 & 0.3635 \\ 0.4888 & 0.4528 & 0.3977 & 0.3875 & 0.3680 & 0.3391 \\ 0.4878 & 0.4346 & 0.3703 & 0.3625 & 0.3436 & 0.3179 \end{bmatrix}$$

Dari kedua nilai a_k dan b_k tersebut didapatkan hasil akhir dari *Guided Flter* ketiga yaitu :

$$q_3 = \begin{bmatrix} 0.5252 & 0.5265 & 0.5262 & 0.5302 & 0.5389 & 0.4830 \\ 0.5256 & 0.5266 & 0.5248 & 0.5254 & 0.5348 & 0.5353 \\ 0.5247 & 0.5246 & 0.5252 & 0.5283 & 0.5343 & 0.5435 \\ 0.5218 & 0.5241 & 0.5217 & 0.5203 & 0.5328 & 0.5446 \\ 0.5257 & 0.5273 & 0.5259 & 0.4998 & 0.5093 & 0.5328 \\ 0.5352 & 0.5364 & 0.5351 & 0.5080 & 0.4733 & 0.5090 \end{bmatrix}$$

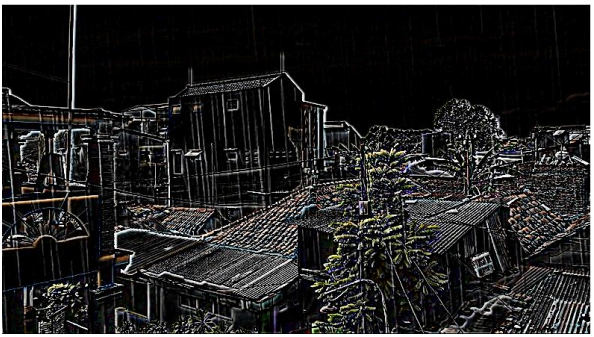
Untuk memberikan gambaran lebih jelas dari proses-proses tersebut, berikut adalah contoh proses-proses dengan menggunakan Gambar 4.3 hingga Gambar 4.12.

Gambar 4.3 Citra *input*

Gambar 4.3 menunjukkan citra *input*. Pada gambar tersebut terlihat *noise-noise* hujan berupa garis-garis berwarna putih. *Noise* tersebut yang akan direduksi.

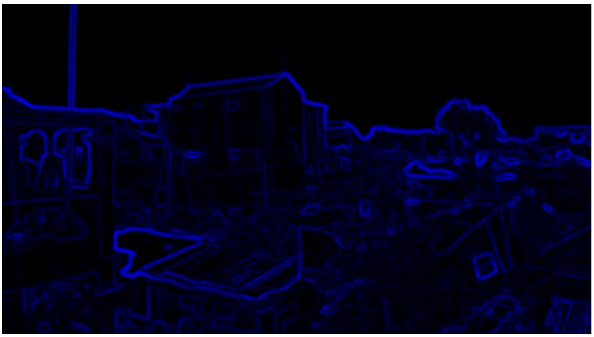
Gambar 4.4 Citra hasil *Guided Filter* I

Setelah diproses *Guided Filter* yang pertama, seperti yang ditunjukkan pada Gambar 4.4, citra menjadi sangat halus dan cenderung kabur karena *Guided Filter* memiliki sifat menghaluskan tepi-tepi citra.



Gambar 4.5 Citra *high frequency*

Citra *input* pada Gambar 4.3 dikurangkan dengan citra hasil *Guided Filter* pertama pada Gambar 4.4 sehingga menghasilkan citra *high frequency* seperti pada Gambar 4.5. Citra *high frequency* hanya mengandung tepi-tepi objek pada citra *input*.

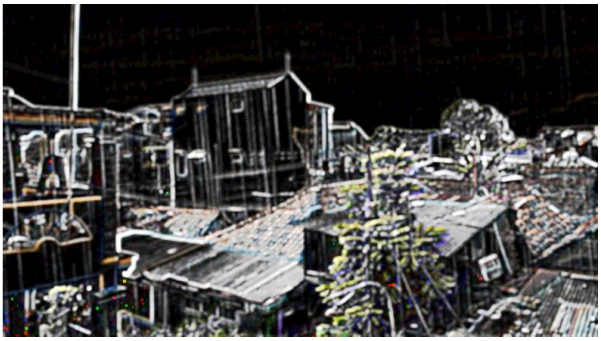


Gambar 4.6 Citra hasil *Sobel Filter*

Tepi-tepi objek pada Gambar 4.4 akan ditajamkan dengan menggunakan *Sobel Filter*. Hasil dari *Sobel Filter* hanya mengandung tepi-tepi objek pada citra hasil *Guided Filter* pertama. Hasil *Sobel Filter* bisa dilihat pada Gambar 4.6

Gambar 4.7 Citra hasil *Edge Enhancement*

Gambar 4.6 kemudian diproses kembali untuk menghasilkan citra RGB sehingga didapatkan citra hasil dari *Edge Enhancement* yang ditunjukkan pada Gambar 4.7. Pada langkah ini, citra masih *blurry* tetapi tingkatnya sudah berkurang.



Gambar 4.8 Citra hasil *Guided Filter* II

Guided Filter kedua dilakukan dengan Gambar 4.5 sebagai input dan Gambar 4.7 sebagai acuan. Karena citra *input* berupa citra *high frequency*, maka dihasilkan citra yang juga hanya mengandung tepi-tepi objek pada citra hasil *Edge Enhancement*.

Gambar 4.9 Citra *Recovery*

Gambar 4.8 kemudian dikembalikan lagi menjadi RGB sehingga menghasilkan citra *recovery* seperti pada Gambar 4.9. Citra ini masih *blurry* sehingga belum menghasilkan hasil maksimal.

Gambar 4.10 Citra *Clear*

Gambar 4.9 diproses agar mendapat citra *clear* seperti pada Gambar 4.10. Pada tahap ini, citra sudah tidak *blurry* dan *noise* hujan juga sudah berkurang. Citra *clear* ini menjadi hasil akhir dari proses *Guided Filter* kedua.

Gambar 4.11 Citra hasil *Weighted Summation*

Proses *Weighted Summation* dilakukan pada Gambar 4.10 untuk menyetarakan *noise* yang masih terlihat pada citra *clear*. Hasilnya ditunjukkan pada Gambar 4.11

Gambar 4.12 Citra hasil *Guided Filter* III

Kemudian dilakukan proses *Guided Filter* ketiga dengan Gambar 4.10 sebagai citra *input* dan Gambar 4.11 sebagai citra acuan. Hasil yang didapat, seperti pada Gambar 4.12, citra kembali menjadi *blurry* karena, seperti yang terjadi pada *Guided Filter* pertama, *Guided Filter* memiliki sifat menhaluskan tepi-tepi objek pada citra.

4.2. Implementasi Program

Program yang telah dirancang sedemikian rupa, kemudian diimplementasikan ke dalam bahasa komputer. Bahasa yang digunakan pada program ini adalah bahasa pemrograman Java dalam IDE Netbeans dengan bantuan *library* OpenCV 3.4.2. Proses yang dilakukan pada saat implementasi program adalah sebagai berikut :

a. Proses input video

Hal pertama yang dilakukan pada program ini adalah menginputkan video dengan *noise* hujan. Video yang diinputkan juga sebaiknya diaur ulang ukurannya agar memiliki ukuran yang seragam. Penginputan video dapat dilakukan dengan cara berikut :

```
Mat imageArray =
Imgcodecs.imread("hujan.jpg");
Mat input = new Mat();
```

```
Size sz = new Size(600,500);
Imgproc.resize(imageArray,     input,
sz);
```

b. Ekstraksi video menjadi frame

Setelah video diinputkan, video diekstraksi menjadi frame-frame dengan cara berikut :

```
public static Mat VideotoFrame(String
filename){
      VideoCapture    cap    =    new
      VideoCapture();
      String input = filename;
      String output = "output";
      cap.open(input);
      int    videolength    =    (int)
      cap.get(Videoio.CAP_PROP_FRAME_
      COUNT);
      int       fps       =       (int)
      cap.get(Videoio.CAP_PROP_FPS);
      int    framenumber    =    (int)
      cap.get(Videoio.CAP_PROP_POS_FR
      AMES);
      Mat frame = new Mat();
      if (cap.isOpened()){

      System.out.println("Number   of
      frames : "+videolength);

      System.out.println(fps+" fps");

      System.out.println("converting
      video...");
      cap.read(frame);

      while(framenumber<=videolength)
      {
```

```
        Imgcodecs.imwrite(output+"/"+fr
        amenumber+".jpg", frame);
                        framenumber++;
        }
        cap.release();

        System.out.println(videolength+
        " frames extracted");
        }
        else
        System.out.println("Fail");
        return frame;
    }
```

Rancangan program reduksi *noise* hujan dengan menggunakan metode *Guided Filter* ini juga dilakukan pada proses ekstraksi video.

c. Proses menampilkan video

Video yang sudah diproses susai dengan yang dijabarkan pada perancangan program akan ditampilkan dengan cara sebagai berikut :

```
public       static       VideoCapture
video(String video){
          Mat frame = new Mat();
        VideoCapture   camera   =   new
        VideoCapture(video);
        JFrame     jframe     =     new
        JFrame("Video");
        jframe.setDefaultCloseOperation
        (JFrame.EXIT_ON_CLOSE);
        JLabel vidpanel = new JLabel();
        jframe.setContentPane(vidpanel)
        ;
        jframe.setVisible(true);
        while (true){
        if (camera.read(frame)){
```

```
        ImageIcon     image     =     new
        ImageIcon(Mat2BufferedImage(fra
        me));
        vidpanel.setIcon(image);
        vidpanel.repaint();
        }
        }
    }
```

Method Mat2BufferedImage digunakan untuk merubah matriks menjadi citra. Method tersebut dapat ditulis seperti berikut :

```
public      static      BufferedImage
Mat2BufferedImage(Mat m){
      int            type           =
      BufferedImage.TYPE_BYTE_GRAY;
      if (m.channels() > 1){
      type                          =
      BufferedImage.TYPE_3BYTE_BGR;
      }
      int         bufferSize        =
      m.channels()*m.cols()*m.rows();
      byte[] b = new byte[bufferSize];
      m.get(0, 0, b);
      BufferedImage   image   =   new
      BufferedImage(m.cols(),
      m.rows(), type);
      final   byte[]   targetPixels   =
      ((DataBufferByte)
      image.getRaster().getDataBuffer
      ()).getData();
      System.arraycopy(b,           0,
      targetPixels, 0, b.length);
      return image;
    }
```

d. Proses *Guided Filter*

Guided Filter adalah proses inti pada Tugas Akhir ini. Berikut merupakan penulisan proses *Guided Filter* pada bahasa pemrograman Java :

```
public static Mat GuidedImageFilter(Mat
I, Mat p, int r, double eps) {
      int rows = I.rows();
      int cols = I.cols();
      // N = boxfilter(ones(hei, wid), r);
      % the size of each local patch;
      N=(2r+1)^2 except for boundary
      pixels.
      Mat N = new Mat();
      Imgproc.boxFilter(Mat.ones(rows,
      cols, I.type()), N, -1, new Size(r,
      r));
      //boxFilter(Mat src, Mat dst, int
      ddepth, Size ksize)
      //smoothes an image using box filter
      //ones(int rows, int cols, int type)
      //return an array of all 1's of the
      specified size and type

      // mean_I = boxfilter(I, r) ./ N;
      Mat mean_I = new Mat();
      Imgproc.boxFilter(I,  mean_I,  -1,
      new Size(r, r));
      // mean_p = boxfilter(p, r) ./ N
      Mat mean_p = new Mat();
      Imgproc.boxFilter(p,  mean_p,  -1,
      new Size(r, r));
      // mean_Ip = boxfilter(I.*p, r) ./
      N;
      Mat mean_Ip = new Mat();
      Imgproc.boxFilter(I.mul(p),
      mean_Ip, -1, new Size(r, r));
      // cov_Ip = mean_Ip - mean_I .*
      mean_p; % this is the covariance of
      (I, p) in each local patch.
```

```
Mat cov_Ip = new Mat();
Core.subtract(mean_Ip,
mean_I.mul(mean_p), cov_Ip);
//subtract(Mat src1, Mat src2, Mat
dst)
//calculates the per-element
difference between two arrays or
array and a scalar

// mean_II = boxfilter(I.*I, r) ./
N;
Mat mean_II = new Mat();
Imgproc.boxFilter(I.mul(I),
mean_II, -1, new Size(r, r));
// var_I = mean_II - mean_I .*
mean_I;
Mat var_I = new Mat();
Core.subtract(mean_II,
mean_I.mul(mean_I), var_I);
// a = cov_Ip ./ (var_I + eps); %
Eqn. (5) in the paper;
Mat a = new Mat();
Mat c = new Mat();
Core.add(var_I, new Scalar(eps),
c);
Core.divide(cov_Ip, c, a);
//b = mean_p - a .* mean_I; % Eqn.
(6) in the paper;
Mat b = new Mat();
Core.subtract(mean_p,
a.mul(mean_I), b);
// mean_a = boxfilter(a, r) ./ N;
Mat mean_a = new Mat();
Imgproc.boxFilter(a, mean_a, -1,
new Size(r, r));
Core.divide(mean_a, N, mean_a);
// mean_b = boxfilter(b, r) ./ N;
```

```
        Mat mean_b = new Mat();
        Imgproc.boxFilter(b,  mean_b,  -1,
        new Size(r, r));
        Core.divide(mean_b, N, mean_b);
        // q = mean_a .* I + mean_b; % Eqn.
        (8) in the paper;
        Mat q = new Mat();
        Core.add(mean_a.mul(I), mean_b, q);
        //    q.convertTo(q,
        CvType.CV_32F);
        return q;
}

public              static              Mat
GuidedImageFilterRGB(Mat I, Mat p, int r,
double eps){
        List<Mat>     guidance     =     new
        ArrayList<>();
        List<Mat>      input      =      new
        ArrayList<>();
        Core.split(p, input);
        Core.split(I, guidance);
        Mat             outputR             =
        GuidedImageFilter(guidance.get(0),
        input.get(0), r, eps);
        Mat                          outputG=
        GuidedImageFilter(guidance.get(1),
        input.get(1), r, eps);
        Mat             outputB             =
        GuidedImageFilter(guidance.get(2),
        input.get(2), r, eps);
        Mat output = new Mat();
        Core.merge(new
        ArrayList<>(Arrays.asList(outputR,
        outputG, outputB)), output);
        return output;
}
```

e. Hitung PSNR

PSNR atau *Peak Signal to Noise Ratio* digunakan untuk membandingkan hasil akhir dengan video asli tanpa *noise* hujan. Koding dari menghitung PSNR adalah sebagai berikut :

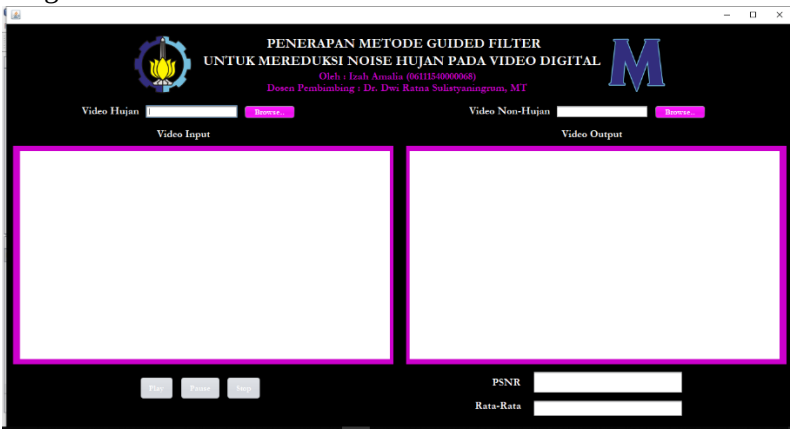
```
public static double
printPSNR(BufferedImage im1,
BufferedImage im2) {
      assert(
      im1.getType() == im2.getType()
      &&      im1.getHeight()      ==
      im2.getHeight()
      &&       im1.getWidth()       ==
      im2.getWidth());

      double mse = 0;
      int width = im1.getWidth();
      int height = im1.getHeight();
      Raster r1 = im1.getRaster();
      Raster r2 = im2.getRaster();
      for (int j = 0; j < height; j++)
      for (int i = 0; i < width; i++)
      mse
      += Math.pow(r1.getSample(i, j,
      0) - r2.getSample(i, j, 0), 2);
      mse   /=   (double)   (width   *
      height);
      System.err.println("MSE = " +
      mse);
      int maxVal = 255;
      double x = Math.pow(maxVal, 2) /
      mse;
      double    psnr    =    10.0    *
      logbase10(x);
      System.err.println("PSNR = " +
      psnr);
```

```
        return psnr;
    }
```

4.3. Tampilan *Interface*

Interface atau antarmuka adalah salah satu yang penting dalam pembuatan program selain implementasi program. Hal pertama yang dilihat oleh pengguna adalah *interface*. Suatu *interface* harus memiliki sifat *user compatibility* atau bisa digunakan dengan baik oleh pengguna. Gambar 4.3 merupakan tampilan *interface* dari program reduksi *noise* hujan pada video digital :



Gambar 4.13. Tampilan interface

Pada program ini, ada beberapa tombol dan kolom. Kolom "Video Hujan" digunakan untuk memasukkan *input* video dengan *noise* hujan yang dapat dicari dengan menekan tombol "Browse.." Dengan cara yang sama, kolom "Video Non Hujan" digunakan untuk memasukkan *input* video tanpa *noise* hujan yang akan digunakan untuk menghitung PSNR video. Pada saat tombol "Play" ditekan, maka video *input* akan ditampilkan pada kolom "Video Input" dan video yang telah diproses akan

ditampilkan pada kolom “Video Output”. Selain itu, akan terlihat PSNR setiap *frame* pada kolom “PSNR.” Setelah tombol “Stop” ditekan, maka video akan berhenti, baik itu video input maupun video output, dan akan ditampilkan rata-rata PSNR video tersebut pada kolom “Rata-rata.”

BAB V
UJI COBA DAN ANALISA HASIL

Bab ini menjabarkan mengenai data uji coba yang digunakan dan analisa hasil yang didapat dari data uji coba tersebut.

5.1. Data Uji Coba

Suatu program pasti membutuhkan data untuk menjalankannya. Pada program reduki *noise* hujan dengan menggunakan metode *Guided Filter* ini menggunakan beberapa data berupa video dengan format .mp4 dan memiliki *frame rate* 24 fps, yaitu :

a. Video dengan objek diam
b. Video dengan objek bergerak

Video-video tersebut mengandung *noise* hujan buatan agar lebih mudah untuk menghitung perbandingan antara video asli tanpa *noise* hujan dan video yang telah diproses untuk mereduksi *noise* hujan. Cara yang digunakan untuk membuat *noise* hujan buatan yaitu dengan mengunduh video *rain overlay* di YouTube dan menambahkan video tersebut ke dalam video asli dengan menggunakan program Adobe Premier Pro. Tipe-tipe hujan yang digunakan juga bermacam-macam, yaitu :

a. Hujan deras dengan butiran hujan yang panjang dan memiliki intensitas yang rapat
b. Hujan sedang dengan butiran hujan yang panjang dan memiliki intensitas yang sedang
c. Hujan gerimis dengan butiran hujan berupa titik-titik dan memiliki intensitas yang sedang.

Dari syarat-syarat diatas, didapatkan enam video yang digunakan sebagai data uji coba. Video-video tersebut ditunjukkan pada Tabel 5.1.

Tabel 5. 1. Data Uji Coba

No.	Nama File	Screenshot Video	Keterangan
1.	objek diam deras.mp4		Video dengan *noise* hujan deras dan objek diam
2.	objek diam gerimis.mp4		Video dengan *noise* hujan gerimis dan objek diam
3.	objek diam sedang.mp4		Video dengan *noise* hujan sedang dan objek diam
4.	objek gerak deras.mp4		Video dengan *noise* hujan deras dan objek bergerak

No.	Nama File	Screenshot Video	Keterangan
5.	objek gerak gerimis.mp4		Video dengan *noise* hujan gerimis dan objek bergerak
6.	objek gerak sedang.mp4		Video dengan *noise* hujan sedang dan objek bergerak

5.2. Hasil Uji Coba

Data-data pada Tabel 5.1 diproses seperti yang sudah dijelaskan pada bab sebelumnya. Dari proses tersebut, didapatkan hasil yang bervariasi. Sub-bab ini menjelaskan hasil yang didapat.

Video-video diproses dengan r atau radius *square window* dan ϵ atau parameter regularisasi yang berbeda, yaitu r=2, 4, dan 8, serta ϵ=0,01, 0,04, dan 0,16. Nilai-nilai tersebut didapatkan dari jurnal Kaiming He yang berjudul *Guided Image Filter*. Masing-masing nilai r dan ϵ dihitung PSNR untuk *guided filter* pertama, *guided filter* kedua, dan *guided filter* ketiga. *Guided filter* pertama didapatkan pada saat program berhenti pada saat proses *guided filter* pertama, begitu juga dengan *guided filter* kedua dan ketiga. Dari hasil ketiga *guided filter* tersebut, dicari nilai PSNR terbesar untuk mendapatkan kinerja terbaik *guided filter* pada video tersebut. PSNR video

didapatkan dari perhitungan rata-rata dari semua *frame* yang telah diproses.

Selanjutnya, setelah mendapat hasil yang paling maksimal, dihitung PSNR dengan membandingkan video *input* dan video *output* untuk membuktikan apakah sudah berhasil mereduksi *noise*. Semakin kecil PSNR, maka semakin bagus reduksi *noise* hujannya.

Selain penilaian secara kuantitatif dengan menggunakan PSNR, dilakukan juga penilaian secara kualitatif atau dengan visual. Penilaiannya yaitu :

a. Baik : *Noise* hujan sudah tereduksi dengan baik dan kualitas video baik.
b. Cukup : *Noise* hujan masih banyak terlihat dan kualitas video baik.
c. Kurang : Kualitas video *blurry* sehingga tidak terlihat apakah *noise* hujan sudah tereduksi atau belum.

5.2.1. Objek diam dan *noise* hujan deras

Hasil dari video objek diam dan *noise* hujan deras dijabarkan dalam Tabel 5.1 di bawah ini :

Tabel 5.2 Perbandingan PSNR pada video objek diam dan *noise* hujan deras

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	21,140	24,78	20,896
2.		0,04	21,140	24,78	20,896
3.		0,16	21,140	24,78	20,896
4.	4	0,01	21,133	24,789	21,106
5.		0,04	21,133	24,789	20,988
6		0,16	21,133	24,789	20,988

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
7.	8	0,01	20,393	24,354	20,284
8.		0,04	20,393	24,354	20,284
9.		0,16	20,393	24,354	20,284
Visual			Kurang	Cukup	Kurang

Tabel 5.2 menunjukkan bahwa untuk video objek diam dan *noise* hujan deras, PSNR terbesar didapatkan pada saat r=4 di *guided filter* kedua, yaitu sebesar 24,789 dB. PSNR tersbut didapatkan untuk nilai ϵ=0,01, 0,04, dan 0,16. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 28,395 dB. Hasil dari *guided filter* kedua pada saat r=4 ditunjukkan pada Gambar 5.1.

Gambar 5.1. Hasil PSNR terbaik untuk objek diam dan *noise* hujan deras

5.2.2. Objek diam dan *noise* hujan gerimis

Hasil dari video objek diam dan *noise* hujan gerimis dijabarkan dalam Tabel 5.2 di bawah ini :

Tabel 5.3 Perbandingan PSNR pada video objek diam dan *noise* hujan gerimis

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	21,297	26,278	21,038
2.		0,04	21,297	26,278	21,038
3.		0,16	21,297	26,278	21,038
4.	4	0,01	21,253	26,193	21,105
5.		0,04	21,253	26,193	21,105
6.		0,16	21,253	26,198	21,109
7.	8	0,01	20,458	25,548	20,349
8.		0,04	20,458	25,548	20,349
9.		0,16	20,458	25,548	20,349
Visual			Kurang	Baik	Kurang

Tabel 5.3 menunjukkan bahwa untuk video objek diam dan *noise* hujan gerimis, PSNR terbesar didapatkan pada saat r=2 di *guided filter* kedua. Nilai ϵ pada kasus ini tidak berpengaruh. Hasil nilai PSNR tersebut adalah 26,278 dB. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 28,77 dB. Hasil tersebut ditampilkan pada Gambar 5.2

Gambar 5.2. Hasil PSNR terbaik untuk objek diam dan *noise* hujan gerimis

5.2.3. Objek diam dan *noise* hujan sedang

Hasil dari video objek diam dan *noise* hujan sedang dijabarkan dalam Tabel 5.3 di bawah ini :

Tabel 5.4 Perbandingan PSNR pada video objek diam dan *noise* hujan sedang

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	21,214	24,94	20,895
2.		0,04	21,214	24,94	20,946
3.		0,16	21,214	24,94	20,895
4.	4	0,01	21,179	24,88	20,956
5.		0,04	21,179	24,88	20,956
6.		0,16	21,179	24,88	20,956
7.	8	0,01	20,437	24,4	20,245
8.		0,04	20,437	24,4	20,245
9.		0,16	20,437	24,4	20,245
Visual			Kurang	Baik	Kurang

Tabel 5.4 menunjukkan bahwa untuk video objek diam dan *noise* hujan sedang, didapatkan PSNR terbesar yaitu 24,94 dB. PSNR tersebut didapatkan pada saat r=2 di *guided filter* kedua. Pada kasus ini, nilai ϵ tidak berpengaruh. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 28,769 dB. Gambar 5.3 menampilkan hasil PSNR tersebut.

Gambar 5.3. Hasil PSNR terbaik untuk objek diam dan *noise* hujan sedang

5.2.4. Objek bergerak dan *noise* hujan deras

Hasil dari video objek bergerak dan *noise* hujan deras dijelaskan dalam Tabel 5.4 di bawah ini :

Tabel 5.5 Perbandingan PSNR pada video objek bergerak dan *noise* hujan deras

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	18,993	18,284	18,835
2.		0,04	18,993	18,284	19,077
3.		0,16	18,993	18,284	19,077
4.	4	0,01	19,376	18,467	19,343
5.		0,04	19,376	18,467	19,343
6.		0,16	19,376	18,467	19,343
7.	8	0,01	19,309	18,563	19,242
8.		0,04	19,309	18,563	19,244
9.		0,16	19,309	18,563	19,242
Visual			Kurang	Cukup	Kurang

Tabel 5.5 menunjukkan bahwa untuk objek bergerak dan *noise* hujan deras, didapatkan PSNR terbesar yaitu 19,376 dB. Nilai tersebut didapatkan pasa saar r=4 di *guided filter* pertama. Pada kasus ini, nilai ϵ tidak berpengaruh untuk nilai PSNR. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 21,82 dB. Hasil dari PSNR tersebut dapat dilihat pada Gambar 5.4.

Gambar 5.4. Hasil PSNR terbaik untuk objek bergerak dan *noise* hujan deras

5.2.5. Objek bergerak dan *noise* hujan gerimis

Hasil dari video objek bergerak dan *noise* hujan gerimis dijelaskan dalam Tabel 5.5 di bawah ini :

Tabel 5.6 Perbandingan PSNR pada video objek bergerak dan *noise* hujan gerimis

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	20,246	18,548	20,08
2.		0,04	20,246	19,862	20,265
3.		0,16	20,246	19,862	20,265
4.	4	0,01	17,695	19,979	20,281
5.		0,04	20,347	19,979	20,281
6.		0,16	20,347	19,979	18,855
7.	8	0,01	19,862	19,947	19,776
8.		0,04	19,862	19,947	19,776
9.		0,16	19,862	19,947	19,776
Visual			Kurang	Baik	Kurang

Tabel 5.6 menunjukkan bahwa untuk objek bergerak dan *noise* hujan gerimis, didapatkan PSNR terbesar pada saat r=4 dan ϵ=0,04 dan 0,16 di *guided filter* pertama. PSNR tersebut yaitu sebesar 20,347 dB. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 21,736 dB. Hasil tersebut dapat dilihat pada Gambar 5.5.

Gambar 5.5. Hasil PSNR terbaik untuk objek bergerak dan *noise* hujan gerimis

5.2.6. Objek bergerak dan *noise* hujan sedang

Hasil dari video objek bergerak dan *noise* hujan sedang dijelaskan dalam Tabel 5.6 di bawah ini :

Tabel 5.7 Perbandingan PSNR pada video objek bergerak dan *noise* hujan sedang

No.	r	ϵ	PSNR Guided Filter I	PSNR Guided Filter II	PSNR Guided Filter III
1.	2	0,01	16,518	18,251	16,233
2.		0,04	16,518	16,185	16,512
3.		0,16	16,518	16,185	16,512
4.	4	0,01	18,963	16,362	16,82
5.		0,04	16,891	16,362	16,82
6.		0,16	16,891	18,855	16,82
7.	8	0,01	17,400	16,593	17,337
8.		0,04	17,400	16,593	17,337
9.		0,16	17,400	16,593	17,337
Visual			Kurang	Baik	Kurang

Tabel 5.7 menunjukkan bahwa untuk video objek bergerak dan *noise* hujan sedang, didapatkan PSNR terbesar dengan nilai 18,963. PSNR tersbut didapatkan pada saat r=4 dan ϵ=0,01 di *guided filter* pertama. Nilai PSNR bila dibandingkan antara output dengan input adalah sebesar 21,722 dB. Hasil tersebut dapat dilihat pada Gambar 5.6.

Gambar 5.6. Hasil PSNR terbaik untuk objek bergerak dan noise hujan sedang

Dari percobaan-percobaan pada Tabel 5.2 sampai Tabel 5.7 PSNR terbesar didapatkan pada saat objek diam dan *noise* hujan gerimis, yaitu sebesar 26,278. PSNR tersebut didapatkan saat r=2 pada proses *guided filter* kedua. PSNR terkecil didapakan pada saat objek bergerak dan *noise* hujan sedang, yaitu sebesar 18,963 dB. PSNR tersebut didapatkan saat r=4 dan ϵ=0,01 pada proses *guided filter* pertama. Sedangkan bila dinilai kualitatif dengan menggunakan visual, video terbaik didapatkan pada saat *guided filter* kedua.

PSNR yang didapatkan pada saat membandingkan video *input* dengan video *output* adalah sebesar 28 dB untuk video dengan objek diam dan 21 dB untuk video dengan objek bergerak.

Rata-rata hasil terbaik didapatkan pada saat *guided filter* kedua. Video yang dihasilkan pada saat *guided filter* kedua adalah video yang hanya mengandung tepi-tepi objek yang kemudian dikembalikan menjadi video RGB, sehingga video yang dihasilkan sudah bagus dan mendekati video asli. Pada

saat *guided filter* ketiga, video yang sudah bagus tersebut diproses kembali dengan *guided filter* sehingga kembali menghasilkan video yang halus dan cenderung sedikit kabur seperti proses *guided filter* pertama karena sifat *guided filter* yang menghaluskan tepi-tepi objek yang terdapat pada video.

BAB VI
SIMPULAN DAN SARAN

Bab ini menjabarkan simpulan yang didapat dari penelitian reduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter* dan saran untuk penelitian di masa mendatang yang berkaitan.

6.1. Simpulan

Simpulan yang didapat setelah melakukan uji coba dan mendapatkan hasil dari penelitian reduksi *noise* hujan dengan menggunakan *Guided Filter* adalah sebagai berikut :

1. Tugas Akhir ini telah berhasil mereduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter*. Hal ini dibuktikan dengan PSNR antara video *input* dan video *output* yaitu sebesar 28 dB untuk video dengan objek diam dan 28 dB untuk video dengan objek bergerak. Implementasi *Guided Filter* dilakukan sebanyak tiga kali. *Guided Filter* pertama mengasilkan citra *low frequency*, dimana citra tersebut sangat halus dan cenderung kabur. Lalu, proses *Guided Filter* yang kedua menghasikan citra yang sudah bagus dan mendekati citra asli. Namun, pada proses *Guided Filter* ketiga, citra yang dihasilkan kembali menjadi sangat halus dan cenderung kabur seperti proses *Guided Filter* pertama. Sehingga, didapatkan video terbaik pada saat proses *Guided Filter* kedua.
2. PSNR yang didapatkan untuk video dengan objek diam adalah dalam interval 20-26 dB. Sedangkan untuk video dengan objek bergerak, didapatkan PSNR dengan interval 16-20 dB. Sehingga dapat disimpulkan bahwa *Guided Filter* dalam penelitian ini kurang optimal untuk video dengan objek bergerak.

6.2. Saran

Adapun saran untuk penelitian selanjutnya yang berhubungan dengan reduksi *noise* hujan pada video digital dengan menggunakan metode *Guided Filter* di masa mendatang adalah sebagai berikut :

1. Melakukan *pre-processing* dengan lebih rinci untuk mendapat tingkat kecerahan setiap *frame* yang setara agar didapatkan hasil yang lebih baik.
2. Menggabungkan metode *Guided Filter* dengan metode lain, seperti *Bilateral Filter*, *Wavelet*, dan sebagainya agar mendapatkan PSNR yang lebih besar. Sehingga hasil yang didapatkan pun lebih akurat untuk menghilangkan *noise* hujan, terlebih untuk video dengan objek bergerak.

DAFTAR PUSTAKA

[1] D. Y. Wardana, "Analisis dan Simulasi Penghilang Hujan Pada CItra DIgital Menggunakan Image Decomposition," 2014.

[2] Najiya C. A., Sreeram S., "Single Image Rain Removal Using Guided Filter," *IJARCSMS*, hal. 135-142, 2015.

[3] Jing Xu dkk, "Removing Rain and Snow in a Single Image using Guided Filter," *IEEE*, hal. 304-307, 2012.

[4] Xianhui Zheng dkk, "Single-Image-Based Rain and Snow Removal Using Multi-Guided Filter," *ICONIP*, hal. 258-265, 2013.

[5] Zheqi Lin, Xuansheng Wang, "Dehazing for Image and Video Using Guided Filter," *Scientific Research*, hal. 123-127, 2012.

[6] S. Khotijah, "Penerapan Transformasi Wavelet Daubechies untuk Reduksi Noise Hujan," 2018.

[7] Kurniawan, "Multimedia Lanjut," *Amikom Yogyakarta*, 2009.

[8] D. Putra, Pengolahan Citra Digital, Yogyakarta: ANDI OFFSET, 2010.

[9] Murinto dkk, "Analisis Perbandingan Metode Intensity Filtering dengan Metode Frequency Filtering sebagai Reduksi Noise Pada Citra Digital," in *Seminar Nasional Aplikasi Tekonologi Informasi 2007*, Jogjakarta, 2007.

[10] Kaiming He dkk, "Guided Image Filtering," 2010.

[11] M. v. Herwijnen, "Weighted Summation".

[12] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, New Jersey: Pearson Education Inc., 2008.

[13] W. K. Pratt, Digital Image Processing, Los Altos: PixelSoft Inc., 2001.

[14] Pulung Nurtantio Andono dkk, Pengolahan Citra Digital, Yogyakarta: CV. Andi Offset, 2017.

[15] Shubham R. Gortarkar, Aarti S. Gaikward, "Filtering of Video using Guided Filter," *International Jornal of Science Technology & Engineering*, vol. 2, no. 11, hal. 848-852, 2016.

[16] Zhenghao Shi dkk, "Weighted Median Guided Filtering Method for Single Image Rain Removal," *EURASIP*, 2018.

LAMPIRAN
Source Code

```
public static Mat GuidedImageFilter(Mat I,
Mat p, int r, double eps) {
//    I.convertTo(I, CvType.CV_64FC1);
//double 64bits double
//    p.convertTo(p, CvType.CV_64FC1);
//
//        Core.divide(p, new Scalar(255,0),
p, CvType.CV_32F);
//        Core.divide(I, new Scalar(255,0),
I, CvType.CV_32F);
      //[hei, wid] = size(I);
      int rows = I.rows();
      int cols = I.cols();
      // N = boxfilter(ones(hei, wid), r); %
the size of each local patch; N=(2r+1)^2
except for boundary pixels.
      Mat N = new Mat();
      Imgproc.boxFilter(Mat.ones(rows, cols,
I.type()), N, -1, new Size(r, r));
        //boxFilter(Mat src, Mat dst, int
ddepth, Size ksize)
        //smoothes an image using box filter
        //ones(int rows, int cols, int type)
        //return an array of all 1's of the
specified size and type

      // mean_I = boxfilter(I, r) ./ N;
      Mat mean_I = new Mat();
      Imgproc.boxFilter(I, mean_I, -1, new
Size(r, r));
      // mean_p = boxfilter(p, r) ./ N
      Mat mean_p = new Mat();
      Imgproc.boxFilter(p, mean_p, -1, new
Size(r, r));
```

```
      // mean_Ip = boxfilter(I.*p, r) ./ N;
      Mat mean_Ip = new Mat();
      Imgproc.boxFilter(I.mul(p), mean_Ip, -
1, new Size(r, r));
      // cov_Ip = mean_Ip - mean_I .*
mean_p; % this is the covariance of (I, p)
in each local patch.
      Mat cov_Ip = new Mat();
      Core.subtract(mean_Ip,
mean_I.mul(mean_p), cov_Ip);
        //subtract(Mat src1, Mat src2, Mat
dst)
        //calculates the per-element
difference between two arrays or array and a
scalar

      // mean_II = boxfilter(I.*I, r) ./ N;
      Mat mean_II = new Mat();
      Imgproc.boxFilter(I.mul(I), mean_II, -
1, new Size(r, r));
      // var_I = mean_II - mean_I .* mean_I;
      Mat var_I = new Mat();
      Core.subtract(mean_II,
mean_I.mul(mean_I), var_I);
      // a = cov_Ip ./ (var_I + eps); % Eqn.
(5) in the paper;
      Mat a = new Mat();
        Mat c = new Mat();
      Core.add(var_I, new Scalar(eps), c);
      Core.divide(cov_Ip, c, a);
      //b = mean_p - a .* mean_I; % Eqn. (6)
in the paper;
      Mat b = new Mat();
      Core.subtract(mean_p, a.mul(mean_I),
b);
      // mean_a = boxfilter(a, r) ./ N;
      Mat mean_a = new Mat();
      Imgproc.boxFilter(a, mean_a, -1, new
Size(r, r));
```

```
      Core.divide(mean_a, N, mean_a);
      // mean_b = boxfilter(b, r) ./ N;
      Mat mean_b = new Mat();
      Imgproc.boxFilter(b, mean_b, -1, new
Size(r, r));
      Core.divide(mean_b, N, mean_b);
      // q = mean_a .* I + mean_b; % Eqn.
(8) in the paper;
      Mat q = new Mat();
      Core.add(mean_a.mul(I), mean_b, q);
//    q.convertTo(q, CvType.CV_32F);
      return q;
    }

    public static Mat
GuidedImageFilterRGB(Mat I, Mat p, int r,
double eps){
        List<Mat> guidance = new
ArrayList<>();
        List<Mat> input = new ArrayList<>();
        Core.split(p, input);
        Core.split(I, guidance);
        Mat outputR =
GuidedImageFilter(guidance.get(0),
input.get(0), r, eps);
        Mat outputG=
GuidedImageFilter(guidance.get(1),
input.get(1), r, eps);
        Mat outputB =
GuidedImageFilter(guidance.get(2),
input.get(2), r, eps);
        Mat output = new Mat();
        Core.merge(new
ArrayList<>(Arrays.asList(outputR, outputG,
outputB)), output);
        return output;
    }
```

```
    public static Mat HighFreq(Mat input,
Mat guided1){
        Mat high = new Mat();
        //I_highfreq = I_input - I_lowfreq
        Core.subtract(input, guided1, high);
        return high;
    }

    public static Mat SobelFilter(Mat
frame){
        Mat grayImage = new Mat();
        Mat detectedEdges = new Mat();
        int scale = 1;
        int delta = 0;
        int ddepth = CvType.CV_16S;
        Mat gradx = new Mat();
        Mat grady = new Mat();
        Mat absgradx = new Mat();
        Mat absgrady = new Mat();

        Imgproc.GaussianBlur(frame, frame,
new Size(3,3), 0, 0, Core.BORDER_DEFAULT);
//        if (frame.channels()==3){
//            Imgproc.cvtColor(frame,
grayImage, Imgproc.COLOR_BGR2GRAY);
//        }
//        else
//            grayImage = frame.clone();
        Imgproc.Sobel(frame, gradx, ddepth,
1, 0);
        Core.convertScaleAbs(gradx,
absgradx);
        Imgproc.Sobel(frame, grady, ddepth,
0, 1);
        Core.convertScaleAbs(grady,
absgrady);
        Core.addWeighted(absgradx, 0.5,
absgrady, 0.5, 0, detectedEdges);
```

```
        return detectedEdges;
    }

    public static Mat EdgeEnhance(Mat
guided1, Mat sobel){
        Mat edgeenhance = new Mat();
        //edgeenhance = guided1 + omega .*
sobel
        Core.multiply(sobel, new
Scalar(0.1), sobel);
        Core.add(guided1, sobel,
edgeenhance);
        return edgeenhance;
    }

    public static Mat Recover(Mat guided2,
Mat guided1){
        Mat recoverimage = new Mat();
        //I_rec = I_guided2 + I_lowfreq
        Core.add(guided2, guided1,
recoverimage);
        return recoverimage;
    }

    public static Mat Clear(Mat recover, Mat
input){
        Mat clearimage = new Mat();
        //I_clr = min(I_rec - I_input)
        Core.min(recover, input,
clearimage);
        return clearimage;
    }

    public static Mat WeightedSummation(Mat
clear, Mat recover){
        Mat refinedimage = new Mat();
        double alpha = 0.8;
```

```
        double satuminalpha = 1 - 0.8;
        //I_ref = alpha*I_clr + (1-
alpha)*I_rec
        Core.multiply(clear, new
Scalar(alpha), clear);
        Core.multiply(recover, new
Scalar(satuminalpha), recover);
        Core.add(clear, recover,
refinedimage);
        return refinedimage;
    }

    public static Mat
WeightedSummationRGB(Mat clear, Mat
recover){
        List<Mat> clr = new ArrayList<>();
        List<Mat> rec = new ArrayList<>();
        Core.split(clear, clr);
        Core.split(recover, rec);
        Mat outputR =
WeightedSummation(clr.get(0), rec.get(0));
        Mat outputG =
WeightedSummation(clr.get(1), rec.get(1));
        Mat outputB =
WeightedSummation(clr.get(2), rec.get(2));
        Mat refinedimage = new Mat();
        Core.merge(new
ArrayList<>(Arrays.asList(outputR, outputG,
outputB)), refinedimage);
        return refinedimage;
    }

    public static double
printPSNR(BufferedImage im1, BufferedImage
im2) {
      assert(
            im1.getType() == im2.getType()
            && im1.getHeight() ==
im2.getHeight()
```

```
            && im1.getWidth() ==
im2.getWidth());

      double mse = 0;
      int width = im1.getWidth();
      int height = im1.getHeight();
      Raster r1 = im1.getRaster();
      Raster r2 = im2.getRaster();
      for (int j = 0; j < height; j++)
            for (int i = 0; i < width; i++)
                  mse
                        +=
Math.pow(r1.getSample(i, j, 0) -
r2.getSample(i, j, 0), 2);
      mse /= (double) (width * height);
      System.err.println("MSE = " + mse);
      int maxVal = 255;
      double x = Math.pow(maxVal, 2) / mse;
      double psnr = 10.0 * logbase10(x);
      System.err.println("PSNR = " + psnr);
      return psnr;
    }

    public static double logbase10(double x)
{
        return Math.log(x) / Math.log(10);
    }

    public static Mat Brightness(Mat input){
        Mat destination = new
Mat(input.rows(), input.cols(),
input.type());
        input.convertTo(destination, -1, 1,
10); //-1,2,25
        return destination;
    }
```

```
    private void setupVideo(){  // setting
label awal warna putih
        Mat localImage = new Mat(new
Size(640, 360), CvType.CV_8UC3, new
Scalar(255, 255, 255));
        Imgproc.resize(localImage,
localImage, new Size(640, 360));
        updateViewIn(localImage);
        updateViewOut(localImage);
    }

    private void updateViewIn(Mat image) {
        vidIn.setIcon(new
ImageIcon(ImageProcessor.Mat2bufferedImage(i
mage)));
    }
    private void updateViewOut(Mat image) {
        vidOut.setIcon(new
ImageIcon(ImageProcessor.Mat2bufferedImage(i
mage)));
    }

    private void
btbrowseActionPerformed(java.awt.event.Actio
nEvent evt) {
        JFileChooser fc = new
JFileChooser("D:\\sasa\\kuliah\\semester
8\\TA\\video\\hasil render\\izah");

fc.setFileSelectionMode(JFileChooser.FILES_O
NLY);
        FileNameExtensionFilter filter = new
FileNameExtensionFilter("Video Files",
"avi", "mp4", "mpg", "mov", "mkv", "mpeg");
        fc.setFileFilter(filter);
        fc.setMultiSelectionEnabled(false);

fc.setAcceptAllFileFilterUsed(false);
```

```
        int option =
fc.showOpenDialog(this);

        if (option ==
JFileChooser.APPROVE_OPTION) {
            videoPath1 =
fc.getSelectedFile().getAbsolutePath();
            vidPath.setText(""+videoPath1);
            capture = new
VideoCapture(videoPath1);
            capture.read(currImage);
            Imgproc.resize(currImage,
currImage, new Size(640, 360));
            updateViewIn(currImage);
            btplay.setEnabled(true);
            //btstop.setEnabled(false);
            //btroi.setEnabled(true);
            //btbiner.setEnabled(true);
        }
    }

    private void
btplayActionPerformed(java.awt.event.ActionE
vent evt) {
        if(isPaused){
            isPaused = false;
            btplay.setEnabled(false);
            btpause.setEnabled(true);
            btstop.setEnabled(true);
            btbrowse.setEnabled(false); //
set true hanya di btstop

            if (pause == 1){
                pause = 0;
            }
            else{
```

```
                capture = new
VideoCapture(videoPath1);
//                capture2 = new
VideoCapture(videoPath2);
                runthread = new RunThread();
                runthread.runnable = true;
                Thread thread1 = new
Thread(runthread);
                thread1.setDaemon(true);
                thread1.start();
            }
        }
    }

    private void
btpauseActionPerformed(java.awt.event.Action
Event evt) {
        if (!isPaused){
            isPaused = true;
            btplay.setEnabled(true);
            btpause.setEnabled(false);
            pause = 1;
        }
    }

    private void
btstopActionPerformed(java.awt.event.ActionE
vent evt) {
        btbrowse.setEnabled(true);
        btplay.setEnabled(true);
        btpause.setEnabled(false);
        isPaused = true;
        pause = 0;
        runthread.runnable = false;
        capture.release();
    }
```

```
    private void
btbrowse1ActionPerformed(java.awt.event.Acti
onEvent evt) {
        // TODO add your handling code here:
        JFileChooser fc = new
JFileChooser("D:\\sasa\\kuliah\\semester
8\\TA\\video");

fc.setFileSelectionMode(JFileChooser.FILES_O
NLY);
        FileNameExtensionFilter filter = new
FileNameExtensionFilter("Video Files",
"avi", "mp4", "mpg", "mov", "mkv", "mpeg");
        fc.setFileFilter(filter);
        fc.setMultiSelectionEnabled(false);

fc.setAcceptAllFileFilterUsed(false);

        int option =
fc.showOpenDialog(this);

        if (option ==
JFileChooser.APPROVE_OPTION) {
            videoPath2 =
fc.getSelectedFile().getAbsolutePath();
            vidPath1.setText(""+videoPath2);
            capture2 = new
VideoCapture(videoPath2);
            capture2.read(videoinput);
//            BufferedImage inputshow =
ImageProcessor.Mat2bufferedImage(videoinput)
;
//            BufferedImage guided3show =
ImageProcessor.Mat2bufferedImage(guided3);
//
psnr.setText(""+printPSNR(guided3show,inputs
how));
```

```
        }
    }

    private void
psnrActionPerformed(java.awt.event.ActionEve
nt evt) {
        // TODO add your handling code here:
    }

    private void
vidPathActionPerformed(java.awt.event.Action
Event evt) {
        // TODO add your handling code here:
    }

    public class RunThread implements
Runnable{
        //variabel
        protected volatile boolean runnable
= false;
        int framecount = 1;
        double psnrcount = 0;
        double mean = 0;

        public void run() {
            if (capture.isOpened()){
                while(runnable){
                    if(!isPaused){

capture.read(currImage);

capture2.read(videoinput);

if(!currImage.empty()){

System.out.println("\n\nframe ke-" +
framecount);
```

```
Imgproc.resize(currImage, currImage, new
Size(640, 360));
                              Mat img =
currImage.clone();
//
Imgproc.cvtColor(img, img,
Imgproc.COLOR_BGR2GRAY);
                              Mat guided =
GuidedImageFilterRGB(img, img, 4, 0.01);
                              Mat high =
HighFreq(img, guided);
                              Mat sobel =
SobelFilter(guided);
                              Mat edge =
EdgeEnhance(guided, sobel);
                              Mat guided2 =
GuidedImageFilterRGB(edge, high, 4, 0.01);
//pake edge bukan sobel
                              Mat recover =
Recover(guided2, guided);
                              Mat clear =
Clear(recover, img);
                              Mat weightedsum
= WeightedSummationRGB(clear, recover);
                              Mat guided3 =
GuidedImageFilterRGB(weightedsum, clear, 4,
0.01);
                              Mat sobel2 =
SobelFilter(guided3);
                              Mat edge2 =
EdgeEnhance(guided3, sobel2);
                              Mat clear2 =
Clear(guided3, img);
                              Mat weightedsum2
= WeightedSummationRGB(clear, guided3);
```

```
                                Mat brightness =
Brightness(weightedsum);
//                                BufferedImage
inputshow =
ImageProcessor.Mat2bufferedImage(videoinput)
;
//                                BufferedImage
clear2show =
ImageProcessor.Mat2bufferedImage(brightness)
;
//
psnr.setText("\n\nframe ke-"+framecount+"
"+printPSNR(clear2show,inputshow));



if(!videoinput.empty()){

Imgproc.resize(videoinput, videoinput, new
Size(640, 360));

BufferedImage inputshow =
ImageProcessor.Mat2bufferedImage(videoinput)
;

BufferedImage brightshow =
ImageProcessor.Mat2bufferedImage(guided);

psnr.setText("\n\nframe ke-"+framecount+"
"+printPSNR(brightshow,inputshow));
                                    psnrcount =
psnrcount+printPSNR(brightshow,inputshow);
                                }
                                else{
                                    break;
                                }
//
updateViewIn(currImage);
```

```
//
updateViewOut(weightedsum);

Imgcodecs.imwrite("frame
"+framecount+".jpg", guided);

                        } else {

btplay.setEnabled(false);
                            break;
                        }
                        mean =
psnrcount/framecount;
//
meanpsnr.setText(""+mean);

System.out.println("frame "+framecount+" :
"+mean);
                        framecount++;
                    }
                }
            }
            meanpsnr.setText(""+mean);
        }

    }

    static class ImageProcessor{
        public static BufferedImage
Mat2bufferedImage(Mat m){
            int type =
BufferedImage.TYPE_BYTE_GRAY;
            if ( m.channels() > 1 )
                type =
BufferedImage.TYPE_3BYTE_BGR;
            int bufferSize =
m.channels()*m.cols()*m.rows();
```

```
            byte [] b = new
byte[bufferSize];
            m.get(0,0,b); // get all the
pixels
            BufferedImage image = new
BufferedImage(m.cols(),m.rows(), type);
            final byte[] targetPixels =
((DataBufferByte)
image.getRaster().getDataBuffer()).getData()
;
            System.arraycopy(b, 0,
targetPixels, 0, b.length);
            return image;
        }
    }
```

TENTANG PENULIS

Penulis memiliki nama lengkap Izah Amalia. Lahir di Surabaya pada hari Sabtu tanggal 22 November 1997. Penulis menempuh pendidikan di SD Muhammadiyah 6 Surabaya (2003-2009), SMP Negeri 1 Surabaya (2009-2012), dan SMA Negeri 2 Surabaya (2012-2015). Penulis memiliki hobi menonton film dan membaca novel. Cita-cita pribadi penulis adalah memiliki novel atau buku karangan sendiri. Penulis aktif sebagai Staff *External Affair Department* HIMATIKA ITS 2016-2017 dalam Divisi Media dan Informasi. Di tahun ketiga kuliah, penulis menjabat sebagai *Head of Media and Information Department* HIMATIKA ITS 2017-2018. Selain di HIMATIKA ITS, penulis juga aktif di UKM Taekwondo. Di Departemen Matematika ini, penulis mengambil rumpun Ilmu Komputer dan ingin menjadi Analis Sistem setelah lulus nanti. Untuk informasi, kritik, atau saran lebih lanjut bisa disampaikan melalui *e-mail* penulis di izahamalia09@gmail.com.