Implementasi *Focus Stacking* Pada Citra Menggunakan Algoritma Pendeteksian Tepi

Muhammad Fahmi Irfan - 13520152
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13520152@std.stei.itb.ac.id

***Abstrak*—*Focus stacking* merupakan salah satu teknik yang umum digunakan pada fotografi. *Focus stacking* ialah teknik menggabungkan beberapa citra dengan *depth of field* yang berbeda menjadi suatu citra dengan cakupan *depth of field* yang luas. Penelitian ini bertujuan untuk melakukan eksplorasi pengimplementasian *focus stacking* menggunakan algoritma pendeteksian tepi, yaitu menggunakan Operator Sobel. Terdapat tiga buah eksplorasi yang dilakukan, yaitu *focus stacking* tanpa penentuan titik fokus, *focus stacking* dengan penentuan titik fokus tanpa pembobotan, dan *focus stacking* dengan penentuan titik fokus dan dengan pembobotan. Ketiga implementasi *focus stacking* memiliki kelebihan dan kekurangannya masing-masing, terutama pada ketajaman dan kehalusan citra.**

***Kata kunci*—*focus stacking*, pendeteksian tepi, Operator Sobel**

I. PENDAHULUAN

Di era modern ini, perkembangan teknologi yang sangat pesat memungkinkan munculnya alat-alat yang membantu kehidupan manusia yang tidak terpikirkan sebelumnya. Salah satu bidang yang menikmati kemajuan teknologi adalah bidang fotografi, di mana teknik pengolahan citra terus berkembang untuk meningkatkan kualitas visual dan memberikan solusi bagi tantangan-tantangan khusus yang dihadapi dalam fotografi, terutama pada bidang makro dan mikro.

Fotografi makro dan mikro sering kali melibatkan objek-objek dengan struktur kompleks dan ukuran yang sangat kecil. Pada fotografi semacam ini, kendala kedalaman bidang seringkali menjadi tantangan utama, di mana fokus hanya dapat ditempatkan pada bagian-bagian tertentu dari objek, sedangkan bagian lainnya menjadi kurang tajam atau kabur. Untuk mengatasi masalah ini, teknik *focus stacking* telah menjadi solusi yang umum digunakan.

Focus stacking melibatkan penggabungan beberapa citra dengan fokus berbeda menjadi satu gambar, menciptakan kesan kedalaman bidang yang lebih besar. Dalam konteks ini, pendeteksian tepi pada citra menjadi penting, karena tepi objek menyimpan informasi kritis tentang perubahan intensitas dan kontras yang dapat digunakan untuk menentukan bagian-bagian yang perlu diberikan fokus lebih tinggi.

Dalam penelitian ini, fokus utama adalah mengimplementasikan teknik *focus stacking* pada citra dengan memanfaatkan algoritma pendeteksian tepi. Algoritma ini dipilih karena kemampuannya untuk mengidentifikasi perubahan signifikan dalam intensitas citra, yang dapat memberikan informasi yang berharga untuk meningkatkan kualitas hasil *focus stacking*. Dengan menggabungkan teknologi *focus stacking* dan algoritma pendeteksian tepi, diharapkan penelitian ini dapat memberikan kontribusi pada perkembangan teknik pengolahan citra, khususnya dalam meningkatkan ketajaman dan kedalaman visual pada fotografi makro dan mikro.

II. LANDASAN TEORI

A. *Depth of Field*

Depth of field menyatakan jarak objek pada suatu citra yang memiliki fokus terbaik. Keberadaan *depth of focus* disebabkan lensa yang digunakan kamera memiliki suatu titik fokus. Ketika objek berada pada titik fokus kamera, hasil citra yang berhasil diambil oleh kamera akan memiliki ketajaman terbaiknya pada objek tersebut.

B. *Focus Stacking*

Focus Stacking merupakan suatu teknik fotografi untuk menghasilkan citra dengan *depth of field* yang meluas. *Focus stacking* biasa digunakan pada citra yang memiliki cakupan *depth of field* yang lebih sempit dari yang diharapkan. Umumnya, *focus stacking* digunakan pada citra lanskap dan citra makro. Pada proses pengambilan citra lanskap, kamera pada umumnya hanya dapat fokus pada salah satu objek. Hal ini cukup merugikan jika objek foto memiliki jarak yang berbeda-beda, sehingga diperlukan *focus stacking* agar hasil citra memiliki fokus yang merata. Pada proses pengambilan citra makro, lensa makro umumnya memiliki fokus yang sangat sempit, sehingga fokus citra hanya pada bagian depan atau bagian belakang objek.

Pada prinsipnya, *focus stacking* melakukan penggabungan dari beberapa citra dengan *depth of field* yang berbeda-beda. Hasil penggabungan tersebut akan menghasilkan sebuah citra yang tajam pada setiap *depth of field* pada citra masukan.

C. Registrasi Citra (*Image Registration*)

Registrasi citra merupakan suatu teknik pengolahan citra yang dapat membandingkan sebuah citra bergerak (*moving image*) terhadap citra diam (*fixed image*), kemudian mengubah posisi dan orientasi fitur-fitur pada citra bergerak sehingga sesuai dengan citra diam.

D. Pendeteksian Tepi (*Edge Detection*)

Tepi adalah perubahan nilai keabuan suatu citra dalam jarak yang cukup singkat. Pendeteksian tepi umumnya digunakan untuk menekankan garis-garis tepi pada suatu citra. Pendeteksian tepi biasanya menggunakan berbagai jenis operator dalam implementasinya, antara lain sebagai berikut.

1. Operator Gradien
   Operator gradien menggunakan pendekatan kalkulus diferensial. Operator gradien menggunakan dua buah *mask* konvolusi berikut.

$$G_x = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad G_y = [-1 \quad 1]$$

2. Operator LoG (Laplacian of Gaussian)
   Operator LoG pada dasarnya ialah melakukan operator Laplacian (operator turunan kedua) terhadap hasil dari operator Gaussian. Operator LoG menggunakan *mask* konvolusi berikut.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3. Operator Sobel
   Operator Sobel merupakan salah satu operator gradient yang dapat digunakan pada pendeteksian tepi. Operator Sobel menggunakan dua buah *mask* konvolusi berikut.

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

4. Operator Prewitt
   Operator Prewitt merupakan salah satu operator gradien yang dapat digunakan pada pendeteksian tepi. Operator Prewitt menggunakan dua buah *mask* konvolusi sebagai berikut.

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

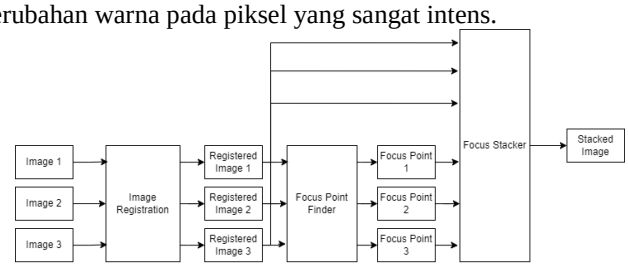
5. Operator Roberts
   Operator Roberts merupakan salah satu operator gradien yang dapat digunakan pada pendeteksian tepi. Operator Roberts menggunakan dua buah *mask* konvolusi sebagai berikut.

$$R_+ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad R_- = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

6. Operator Canny
   Operator Canny merupakan operator gradien yang dapat digunakan pada pendeteksian tepi. Operator Canny dapat menghasilkan citra tepi dengan ketebalan tepi 1 piksel.

III. Implementasi

Tujuan utama dalam implementasi *focus stacking* ialah menyatukan beberapa citra dengan masing-masing fokus berbeda menjadi sebuah citra dengan fokus yang merata. Oleh karena itu, perlu ada mekanisme dalam pendeteksian fokus dari masing-masing citra. Pada hal ini, pendeteksian fokus dilakukan dengan menggunakan algoritma pendeteksian tepi. Hal ini digunakan karena pendeteksian tepi dilakukan dengan cara mencari perubahan piksel tetangga yang drastis. Perubahan piksel tetangga ini juga dapat menyatakan ketajaman warna pada piksel tersebut. Semakin besar perubahan warna pada suatu piksel, maka semakin besar pula ketajaman pada piksel tersebut. Ketajaman pada suatu piksel juga berkorelasi dengan fokus objek pada suatu citra, sehingga titik fokus objek juga dapat dilihat dari daerah yang memiliki perubahan warna pada piksel yang sangat intens.



Gambar III.1 Arsitektur Implementasi *Focus Stacking* yang akan dibangun

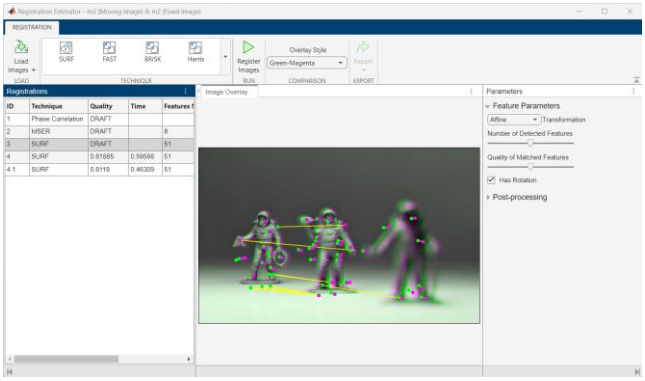
Dari Gambar III.1, terdapat tiga komponen utama dalam implementasi *focus stacking* yang akan dibangun. Ketiga komponen tersebut ialah *image registration, focus point finder*, dan *focus stacker*. Rincian dari ketiga komponen tersebut ialah sebagai berikut.

1. *Image registration*

Pada komponen ini, semua citra akan disesuaikan sehingga fitur-fiturnya memiliki posisi dan orientasi yang sama. Hal ini dilakukan agar memudahkan implementasi *focus stacking*, karena *focus stacking* akan dilakukan per piksel. Jika tidak ada tahap registrasi, akan ada bagian suatu citra yang tidak sesuai dengan citra lain, sehingga akan menghasilkan hasil yang kurang baik.

Algoritma yang digunakan pada *image registration* ialah SURF, dengan transformasi *affine*. Penjelasan lebih lanjut terkait algoritma SURF dengan transformasi *affine* bukan merupakan fokus pada karya tulis ini, sehingga tidak akan

dijelaskan secara rinci. Implementasi *image registration* menggunakan aplikasi *Registration Estimator* yang disediakan oleh Matlab. Tampilan aplikasi tersebut digambarkan pada Gambar III.2.



Gambar III.2 Tampilan Aplikasi *Registration Estimator*

2. *Focus point finder*

Pada komponen ini, dilakukan pencarian titik fokus pada masing-masing citra yang telah dilakukan registrasi. Terdapat beberapa tahap dalam implementasi komponen ini.

Pada tahap pertama, citra akan diubah menjadi citra *grayscale*. Hal ini dilakukan karena tahap-tahap selanjutnya tidak dapat dilakukan menggunakan citra RGB.

Pada tahap kedua, lakukan deteksi tepi menggunakan algoritma deteksi tepi dengan menggunakan operator tertentu. Operator yang digunakan pada deteksi tepi ini ialah Operator Sobel. Dengan melakukan *thresholding* pada suatu nilai tertentu, diperoleh citra biner yang menunjukkan piksel-piksel yang memiliki hasil operasi Operator Sobel tertinggi.

Pada tahap ketiga, lakukan pengisian lubang-lubang pada citra biner, dan kemudian citra biner tersebut disegmentasi. Dari segmen-segmen yang diperoleh, hanya beberapa segmen dengan daerah terluas yang dipertahankan, sementara segmen-segmen kecil lainnya dibuang. Hal ini dikarenakan ada beberapa segmen-segmen kecil lain yang tidak relevan dengan titik fokus pada citra tersebut.

Tahap terakhir dari komponen ini ialah mencari *centroid* dari citra biner yang dihasilkan. Hal ini akan menjadi representasi titik fokus pada masing-masing citra. Titik fokus ini akan memegang peran penting pada komponen selanjutnya.

3. *Focus stacker*

Komponen ini menggunakan semua citra yang telah diregistrasi dan titik-titik fokusnya yang diperoleh dari dua komponen sebelumnya. Terdapat tiga opsi dalam implementasi komponen ini.

Opsi pertama ialah membandingkan hasil operator pada komponen sebelumnya pada masing-masing piksel. Piksel tersebut akan diisi oleh piksel dari citra yang memiliki hasil operasi operator tertinggi. Pada opsi ini, pencarian titik fokus pada masing-masing citra tidak diperlukan.

Opsi kedua ialah membandingkan jarak tiap piksel ke masing-masing titik fokus. Piksel tersebut akan diisi dengan piksel dari citra yang memiliki titik fokus terdekat dengan piksel tersebut.

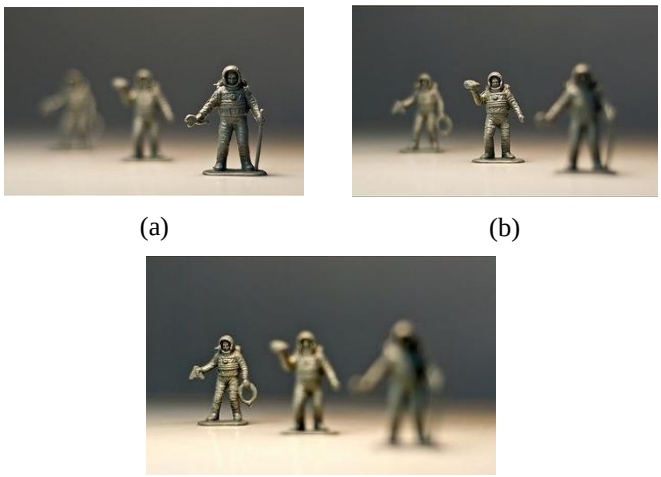
Opsi ketiga ialah melakukan pembobotan pada piksel sesuai jarak piksel tersebut ke masing-masing titik fokus. Citra yang memiliki titik fokus yang lebih dekat pada suatu piksel akan memiliki bobot yang lebih besar pada piksel tersebut dibandingkan citra yang titik fokusnya lebih jauh. Perhitungan bobot pada masing-masing piksel dirumuskan dengan persamaan (1), dengan w(i,j,k) menyatakan bobot pada piksel (i,j) untuk citra ke-k, n menyatakan banyaknya citra masukan, dist(a,b) menyatakan jarak *euclidean* dari titik a ke titik b, dan sum(i,j) merupakan jumlah jarak dari titik (i,j) ke masing-masing titik fokus .

$$w[i,j,k] = \frac{1}{n-1}\left(1 - \frac{dist((i,j), focuspoint[k]))}{sum(i,j))}\right) \quad (1)$$

IV. Hasil dan Pembahasan

A. Citra masukan

Citra masukan yang digunakan ialah tiga buah citra dengan objek yang sama, namun dengan titik fokus yang berbeda-beda. Ketiga citra tersebut dapat dilihat pada Gambar IV.1.



(a) (b)

(c)

Gambar IV.1 Citra masukan dengan (a) titik fokus berada di depan, (b) titik fokus berada di tengah, dan (c) titik fokus berada di belakang

B. *Image Registration*

Pada tahap ini, citra yang menjadi citra diam ialah citra yang fokusnya berada di belakang, yaitu Gambar IV.1(c). Dua citra lainnya menjadi citra bergerak. Kedua citra tersebut dilakukan registrasi terhadap citra diam, sehingga menghasilkan citra yang telah diregistrasi, digambarkan pada Gambar IV.2.


Gambar IV.2 Hasil registrasi citra dengan (a) titik fokus berada di depan, dan (b) titik fokus berada di tengah

C. *Focus point finder*

Pada tahap ini, masing-masing citra akan dicari titik fokusnya. Pencarian titik fokus menggunakan kode di bawah ini

```
mg1 = double(rgb2gray(m1));
grad1 = gradient_operator(mg1, "sobel");
mask1 = grad1 > threshold;
mask1 = imclearborder(mask1);
mask1 = imfill(mask1, 8, 'holes');
mask1_filt = bwareafilt(mask1,
num_of_segments);
stat1 = regionprops(mask1_filt, 'centroid');
cen1 = stat1.Centroid
```

Kode tersebut melakukan pencarian titik fokus pada citra m1, dengan suatu nilai *threshold* dan jumlah segmen tertentu. Hasil dari kode tersebut ialah cen1 yang menyimpan titik fokus dari citra tersebut. Citra kedua dan citra ketiga menggunakan kode yang sama untuk menemukan titik fokus dari masing-masing citra tersebut. Hasil pencarian titik fokus pada ketiga citra tersebut ditunjukkan pada Tabel IV.1.

Tabel IV.1 Hasil pencarian titik fokus

Citra	x	y
Citra 1 (titik fokus di depan)	241.2186	102.5898
Citra 2 (titik fokus di tengah)	79.8134	124.7024
Citra 3 (titik fokus di belakang)	131.6591	94.0851

D. *Focus Stacker*

Pada tahap ini, terdapat tiga buah opsi pengerjaan. Ketiga hasil implementasi opsi tersebut digambarkan pada tabel berikut.

Tabel IV.2 Hasil implementasi *focus stacking*

Opsi	**Hasil**
Opsi pertama (tanpa pencarian titik fokus)	
Opsi kedua (dengan pencarian titik fokus, tanpa pembobotan)	
Opsi ketiga (dengan pencarian titik fokus dan dengan pembobotan)	

Dari ketiga hasil di atas, dapat dilihat bahwa ketiga opsi menghasilkan hasil citra dengan kualitas yang berbeda-beda. Pada opsi kedua pertama, citra yang dihasilkan memiliki ketajaman yang paling baik dibandingkan kedua citra yang dihasilkan menggunakan opsi yang lain. Akan tetapi, citra pertama menghasilkan artefak-artefak yang tidak dibutuhkan. Contohnya ialah pada tepi-tepi objek terdepan seperti pada Gambar IV.3. Pada objek tersebut, terlihat sedikit tepi tambahan yang mengelilingi objek. Hal tersebut disebabkan ketidaksempurnaan dalam registrasi citra, sehingga ada sedikit perbedaan posisi objek pada citra-citra tersebut. Perbedaan posisi tersebut mengakibatkan perbedaan tepi objek tersebut. Karena algoritma yang digunakan pada opsi ini membandingkan hasil operasi operator Sobel pada masing-masing citra, piksel tepi akan sangat mungkin untuk mengisi citra hasil, sehingga seakan-akan muncul tepi bayangan pada implementasi opsi ini.

Gambar IV.3 Objek Terdepan pada Citra 1

Pada opsi kedua, kualitas citra terlihat lebih baik dibandingkan citra pertama, walaupun ketajaman citranya sedikit kurang dibandingkan citra sebelumnya. Hal ini dikarenakan opsi ini pada dasarnya ialah membagi daerah citra berdasarkan titik-titik fokus pada citra masukan. Daerah-daerah tersebut kemudian diisi oleh citra yang memiliki titik fokus pada daerah tersebut. Akan tetapi, kelemahan dari opsi ini ialah pembagian daerah tersebut sangat terlihat. Hal tersebut dikarenakan citra tersebut hanyalah potongan-

potongan dari citra-citra masukan yang memiliki fokus di titik-titik tertentu.

Pada opsi ketiga, kualitas citra yang dihasilkan lebih *smooth* dibandingkan dua citra yang lain. Hal ini dikarenakan pembobotan yang dilakukan dalam pengisian piksel-pikselnya. Akan tetapi, kelemahan dari opsi ini adalah ketajaman hasil citra tidak begitu baik, terutama pada titik-titik fokus pada citra bergerak saat registrasi citra (dalam hal ini, objek depan dan tengah memiliki ketajaman yang kurang baik dibandingkan objek belakang).

Untuk meningkatkan kualitas citra ketiga, telah dilakukan berbagai upaya. Salah satunya ialah penyesuaian algoritma pembobotan. Tiga algoritma lain yang telah dieksperimenkan ialah Persamaan (2), (3), dan (4).

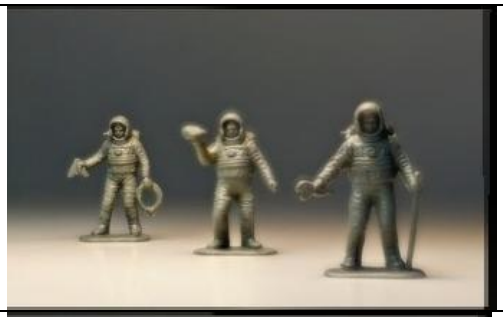
$$w[i,j,k] = \frac{1}{n-1}\left(1 - \frac{dist((i,j), focuspoint[k]))^4}{sum(i,j))}\right) \quad (2)$$

$$w[i,j,k] = \frac{1}{n-1}\left(1 - \frac{dist((i,j), focuspoint[k]))^{16}}{sum(i,j))}\right) \quad (3)$$

$$w[i,j,k] = \frac{1}{n-1}\left(1 - \frac{e^{dist((i,j), focuspoint[k]))}}{sum(i,j))}\right) \quad (4)$$

Hasil eksplorasi menggunakan ketiga algoritma tersebut digambarkan pada Tabel IV.1. Akan tetapi, hasilnya tidak begitu jauh berbeda dengan algoritma pembobotan dengan Persamaan (1).

Tabel IV.3 Tabel eksplorasi algoritma pembobotan

Algoritma Pembobotan	**Hasil**
Persamaan 1	
Persamaan 2	
Persamaan 3	
Persamaan 4	

Selain itu, dilakukan juga percobaan dengan hanya dua buah citra saja. Dalam hal ini, digunakan citra dengan titik fokus berada di depan dan belakang. Hasil *focus stacking* terhadap dua buah citra tersebut menghasilkan hasil yang cukup baik, yang digambarkan pada Gambar IV.4. Hal ini mungkin terjadi karena titik fokus kedua citra tersebut berjauhan, sehingga masing-masing citra tidak begitu mempengaruhi ketajaman pada titik fokus citra yang lain.

Gambar IV.4 Hasil *focus stacking* dengan dua citra

V. KESIMPULAN

Focus stacking dapat diimplementasikan menggunakan algoritma pendeteksian tepi dengan berbagai macam pendekatan. Masing-masing pendekatan memiliki *trade-off* pada aspek kehalusan dan ketajaman citra. Pada pendekatan tanpa pencarian titik fokus, citra yang dihasilkan sangat tajam, namun terdapat banyak derau. Pada pendekatan dengan pencarian titik fokus tanpa pembobotan, citra yang dihasilkan masih cukup tajam walaupun tidak setajam citra pertama, tetapi masih terlihat derau yang cukup krusial. Pada pendekatan dengan pencarian titik fokus dengan pembobotan, citra yang dihasilkan cukup halus, namun ketajamannya tidak sebaik dua citra sebelumnya. Saran dalam pengembangan teknik *focus stacking* ini ialah dengan melakukan identifikasi radius fokus pada tiap citra. Hal ini mungkin dapat meningkatkan kualitas hasil *focus stacking*, karena jarak antar titik fokus berpengaruh pada hasil citra, seperti yang ditunjukkan pada implementasi *focus stacking* dengan dua citra.

UCAPAN TERIMA KASIH

Penulis mengucapkan rasa syukur kepada Allah SWT karena atas berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah berjudul "Implementasi *Focus Stacking* Pada Citra Menggunakan Algoritma Pendeteksian Tepi" dengan tepat waktu. Penulis juga ingin menyampaikan terima kasih kepada keluarga penulis yang selalu memberikan dukungan selama penulis menjalan kehidupan akademik di Institut Teknologi Bandung. Penulis juga ingin menyampaikan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. yang telah membimbing penulis selama pelaksanaan mata kuliah IF4071 Interpretasi dan Pengolahan Citra.

LINK YOUTUBE

Berikut merupakan tautan video youtube makalah ini.
https://youtu.be/tlchTD4T7Hg

REFERENCES

[1] D. Choi, A. Pazylbekova, W. Zhou and P. van Beek, "Improved image selection for focus stacking in digital photography," 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 2761-2765, doi: 10.1109/ICIP.2017.8296785.

[2] McGuinness, C. (2015). Simple Focus Stacking in Python. Diakses di https://github.com/cmcguinness/focusstack

[3] Munir, Rinaldi (2023). 18 – Pendeteksian tepi (edge detection) (Bagian 1). Diakses di https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/18-Pendeteksian-Tepi-Bagian1-2022.pdf

[4] Munir, Rinaldi. 2023. 19 –Pendeteksian tepi (edge detection) (Bagian 2) (Update 2023). Diakses di https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2023-2024/19-Pendeteksian-Tepi-Bagian2-2023.pdf

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2023

Muhammad Fahmi Irfan (13520152)