Perhitungan Jarak Nyata Antara Dua Objek Pada Suatu Foto Dengan Metode Edge Detection

Vincentius Riandaru Prasetyo
JurusanTeknik Informatika
Universitas Surabaya
vincent@staff.ubaya.ac.id

Abstrak – Gambar adalah sebuah artefak yang menggambarkan atau merekam persepsi visual. Gambar juga dapat diartikan sebagai pemetaan objek 3D ke media tertentu. Banyak informasi dapat ditemukan dari foto, misalnya: model kamera, perangkat lunak yang digunakan, kedalaman foto, fokus kamera, dan sebagainya. Ada beberapa informasi yang tidak dapat ditemukan dalam foto, yaitu jarak nyata antara objek dalam foto dan posisi kamera, dan jarak nyata antara 2 objek dalam foto. Sistem yang dibangun berguna untuk menghitung jarak nyata antara objek di foto dengan posisi kamera dan juga jarak nyata antara 2 objek pada suatu foto. Struktur *data tree* digunakan untuk menyimpan nilai-nilai abu-abu (hasil rata-rata nilai merah, hijau, dan biru). *Edge detection* dengan operator Sobel digunakan sebagai metode utama untuk mendapatkan margin (atas, bawah, kiri, dan kanan) dari objek dalam foto.

Kata Kunci: *Edge Detection*, Foto, Sobel, *Tree*.

I. PENDAHULUAN

Citra (gambar) adalah representasi dua dimentasi untuk bentuk-bentuk fisik nyata tiga dimensi. Citra dalam perwujutan dapat bermacam-macam, mulai dari gambar putih pada sebuah foto (yang tidak bergerak) sampai pada gambar warna yang bergerak pada televisi. Proses transformasi dari bentuk tiga dimensi ke bentuk dua dimensi untuk menghasilkan citra akan dipengaruhi oleh bermacam-macam faktor yang mengakibatkan citra penampilan citra suatu benda tidak sama persis dengan bentuk fisiknya [1]. Secara umum, citra berisi 3 elemen warna dasar seperti *Red*, *Green*, dan *Blue* (RGB). Setiap elemen warna dapat dikombinasikan menjadi warna baru. Selain RGB, juga dikenal 4 elemen warna primer seperti *Cyan*, *Magenta*, *Yellow*, dan *Black* (CMYK). CMYK sering digunakan untuk mewarnai pada *printer*.

Media citra yang biasa dikenal dan ditemukan adalah lukisan dan foto. Dalam perkembangannya, lukisan mulai ditinggalkan. Di sisi lain, foto mengalami perkembangan yang pesat. Perkembangan tersebut seiring dengan perkembangan teknologi kamera. Metode atau proses untuk membuat foto disebut fotografi. Foto diciptakan dari pantulan cahaya pada objek.

Banyak informasi yang dapat ditemukan dari sebuah foto original (tanpa ada proses edit sebelumnya), seperti: *bit depth*, *shutter speed*, *lens aperture*, *focal length*, dan sebagainya. Ada beberapa informasi yang tidak dapat ditemukan dalam foto, yaitu jarak nyata antara objek dalam foto dan posisi kamera dengan jarak nyata antara 2 objek dalam foto.

Edge detection adalah salah satu proses ekstraksi fitur yang mengidentifikasi tepian citra, yaitu posisi dimana terjadi perubahan intensitas piksel secara tajam. Tepian dari suatu citra mengandung informasi penting dan mampu merepresentasikan objek-objek yang terkandung dalam citra tersebut meliputi bentuk, ukuran serta tekstur [2]. Operator Sobel merupakan salah satu operator yang digunakan untuk perhitungan pada *edge detection*. Operator Sobel karena menghasilkan hasil ekstraksi paling halus dan memberikan kinerja paling baik dibandingkan operator lain seperti Canny dan Prewitt.

Penelitian dengan menggunakan metode *edge detection* dengan operator sobel sudah pernah dilakukan sebelumnya, salah satunya yang dilakukan oleh Putra, et.al [3]. Pada penelitian tersebut, metode *edge detection* dengan operator Sobel digunakan untuk mengukur tinggi badan manusia. Berbeda dengan penelitian yang dilakukan oleh Putra, et.al [3], penelitian yang dilakukan oleh Utami, et.al [4] menggunakan metode *edge detection* untuk menghitung luas objek pada suatu citra digital. Operator yang digunakan pada penelitian tersebut adalah Canny. Wiyagi dan Mustar [5] pada penelitiannya memanfaatkan *library* OpenCV untuk mendeteksi jarak objek dengan menggunakan kamera tunggal. Jarak objek yang dimaksud pada penelitian tersebut adalah jarak objek terhadap posisi kamera. Objek yang digunakan adalah objek-objek yang bercahaya, seperti lampu LED. Sama seperti Wiyagi dan Mustar [5], penelitian yang dilakukan oleh Kartowisastro [6] dan Haryanti [7] bertujuan untuk menghitung jarak antara posisi kamera dan objek. Akan tetapi, Kartowisastro [6] dan Haryanti [7] menggunakan metode *stereo vision* untuk menghitung jarak tersebut. Selain itu, penelitian oleh Haryanti [7] diimplementasikan pada *mobile phone*, sedangkan Kartowisastro [6] hanya mengimplementasikan pada aplikasi *desktop*.

Berdasarkan permasalahan di atas, sistem yang dibuat berguna untuk menghitung jarak nyata antara objek di foto dan posisi kamera dengan jarak nyata antara 2 objek dalam foto. Jarak tersebut dihitung berdasarkan informasi *focal*

length yang didapatkan dari sebuah foto. Selain itu, sistem yang dibangun juga menggunakan metode *Edge Detection* dengan operator Sobel.

II. METODOLOGI PENELITIAN

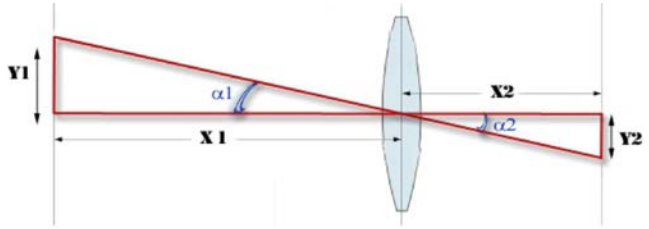
A. Citra (*Image*)

Citra atau *image* adalah angka, dari segi estetika, citra atau gambar adalah kumpulan warna yang bisa terlihat indah, memiliki pola, berbentuk abstrak dan lain sebagainya. Citra dapat berupa foto udara, penampang lintang (*cross section*) dari suatu benda, gambar wajah, hasil tomografi otak, dan lain sebagainya. Dari segi ilmiah, citra adalah gambar 3-dimensi (3D) dari suatu fungsi, biasanya intensitas warna sebagai fungsi *spatial* x dan y. Di komputer, warna dapat dinyatakan, misalnya sebagai angka dalam bentuk skala RGB (*Red, Green, Blue*). Karena citra adalah angka, maka citra dapat diproses secara digital [8].

B. Konsep Foto Digital dan Perbandingan Skala

Dalam fotografi analog, jejak atas obyek yang ditangkap merupakan jejak fisik yang ditampilkan di atas permukaan medium analog (misalnya pelat, film, atau kertas). Sedangkan untuk teknologi digital, jejak fisik yang sifatnya permanen digantikan dengan jejak virtual berupa data elektronis yang disimpan dalam bentuk angka biner atau bit (singkatan dari *Binary Digit*), yakni angka nol dan satu. Kumpulan kode biner itu disimpan dalam bentuk *file*. *File* berisi kombinasi kode biner dapat diterjemahkan kembali sebagai gambar yang dibentuk dari kumpulan titik-titik cahaya (*pixel*) [9]. Masing-masing piksel memiliki intensitas warna merah (*Red*), hijau (*Green*), dan biru (*Blue*) dengan range nilai 0-255.

Terdapat 4 variabel penting yang digunakan untuk menghitung jarak nyata antara objek dengan posisi kamera yaitu: tinggi nyata, jarak nyata antara objek dengan posisi kamera, tinggi objek pada foto, dan jarak fokus kamera. Ilustrasi terbentuknya bayangan pada sebuah foto ditunjukkan pada Gambar 1.



Gambar 1. Ilustrasi Bayangan Objek Pada Foto.

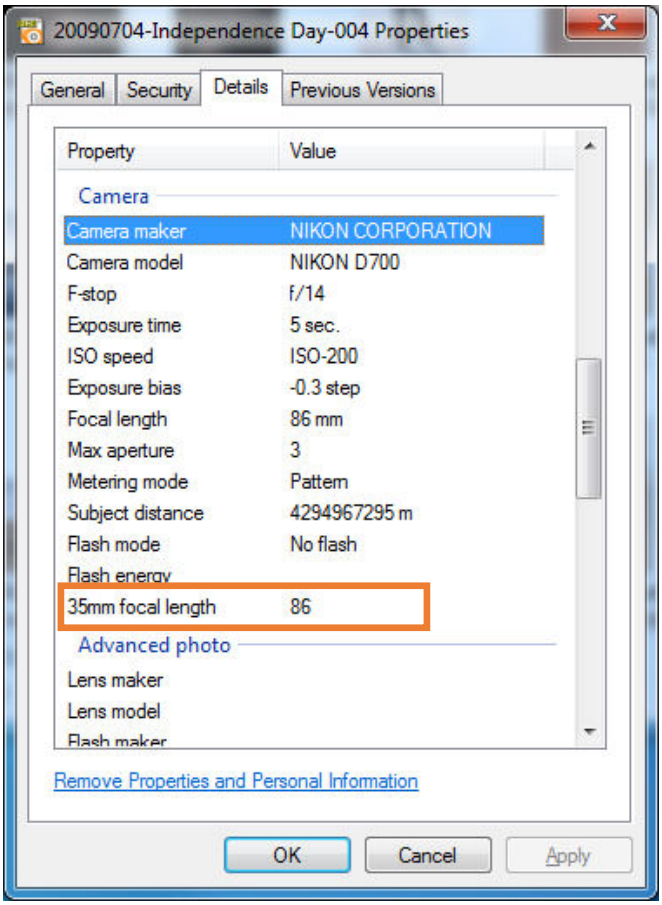
Berdasarkan ilustrasi pada Gambar 1, maka akan didapatkan persamaan (1).

$$\frac{Y1}{X1} = \frac{Y2}{X2} \quad (1)$$

di mana Y_1 adalah tinggi nyata dari objek, X_1 jarak asli antara objek dengan kamera, Y_2 adalah tinggi objek pada foto, dan X_2 adalah jarak fokus kamera. Dengan demikian nilai X_1 dapat dihitung dengan persamaan (2).

$$X1 = \frac{X2}{Y2}\ x\ Y1 \quad (2)$$

Semua foto menyimpan berbagai macam informasi yang disebut EXIF. Untuk dapat menghitung nilai X_1, foto yang digunakan harus foto asli/*original* (tanpa melalui proses *editing*). Foto asli/*original* memiliki nilai X_2 yang dapat dilihat pada EXIF. Nilai X_2 atau fokus kamera dalam EXIF disebut *focal length* seperti yang ditunjukkan oleh Gambar 2.



Gambar 2. *Focal Length* Sebuah Foto *Original*.

Sedangkan untuk nilai Y_2 didapat dengan cara menyeleksi objek pada foto yang akan dihitung jarak nyatanya. Nilai Y_1 didapatkan dengan cara pengukuran secara manual oleh *user*. Jika nilai X_1 pada objek pertama sudah didapatkan, maka jarak objek pertama dan kedua yang sebenarnya dapat dihitung dengan persamaan (3).

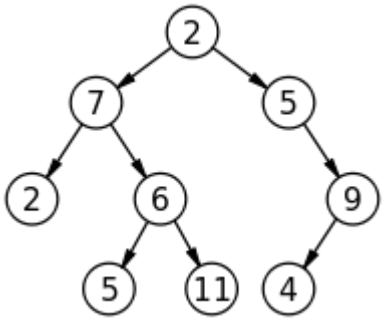
$$Jarak = |X1a - \left(\frac{Y1b\ x\ Y2a\ x\ X1a}{Y1a\ x\ Y2b}\right)| \quad (3)$$

di mana *a* adalah objek pertama dan *b* adalah objek kedua.

C. *Binary Tree*

Binary tree adalah sebuah pohon pencarian yang berisi hanya dua buah cabang yang mana akan dicari simpul sebagai

data yang dibutuhkan. Karakteristik yang dimiliki *binary tree*, yaitu pada sebuah simpul yang merupakan orang tua (*parent*) atau yang disebut juga dengan akar hanya boleh memiliki 0, 1 atau maksimum 2 anak, seperti pada Gambar 3 berikut ini [10].



Gambar 3. *Binary Tree*

Pada penelusuran *binary tree* ini ada kriteria tertentu yang penting yaitu suatu simpul akan diketakkan dalam cabang kiri apabila isi informasi yamg dikirimkan lebih kecil dibanding informasi pada akar, atau sebagai cabang kanan jika isi informasinya dinyatakan lebih besar dari akar. Dengan cara ini pencarian suatu data bisa dilakukan dengan cepat dan mudah.

D. *Edge Detection*

Deteksi tepi (*edge detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek gambar. Suatu titik (x, y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangga. Pendeteksian tepi citra berfungsi untuk memperoleh tepi objek. Deteksi tepi memanfaatkan perubahan nilai intensitas yang drastis pada batas dua area. Jika suatu citra jelas dan tajam maka untuk menentukan letak tepi suatu citra akan lebih mudah, namun jika suatu citra tidak jelas dan mendapatkan gangguan seperti adanya *noise* maka akan timbul kesulitan dalam menentukan letak tepi suatu citra [11].

E. Sobel Operator

Edge detection dapat dilakukan dengan menggunakan beberapa operator perhitungan seperti Prewitt, Sobel, dan Canny. Operator yang sering digunakan karena kelebihan yang dimilikinya adalah Sobel. Selain itu, operator ini juga mudah untuk diimplementasikan.

Operator Sobel merupakan pengembangan dari metode Robert dengan menggunakan HPF (*high pass filter*) yang diberi satu angka nol penyangga. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan pendeteksian tepi [12]. *Gradient* pada operator Sobel dapat dilihat pada persamaan (4).

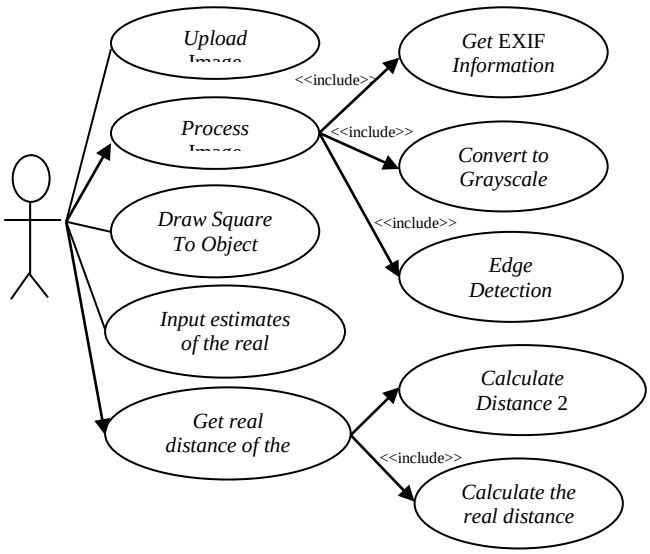
$$Sx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad Sy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (4)$$

III. HASIL DAN PEMBAHASAN

Untuk mempermudah implementasi sistem, maka dibutuhkan analisis dan pembuatan desain sistem. Hal ini bertujuan agar sistem nantinya berjalan dengan baik.

A. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja *user* yang berhak menggunakan fungsi-fungsi tersebut. Gambar 4 di bawah ini memperlihatkan *use case diagram* dari sistem ini [13].



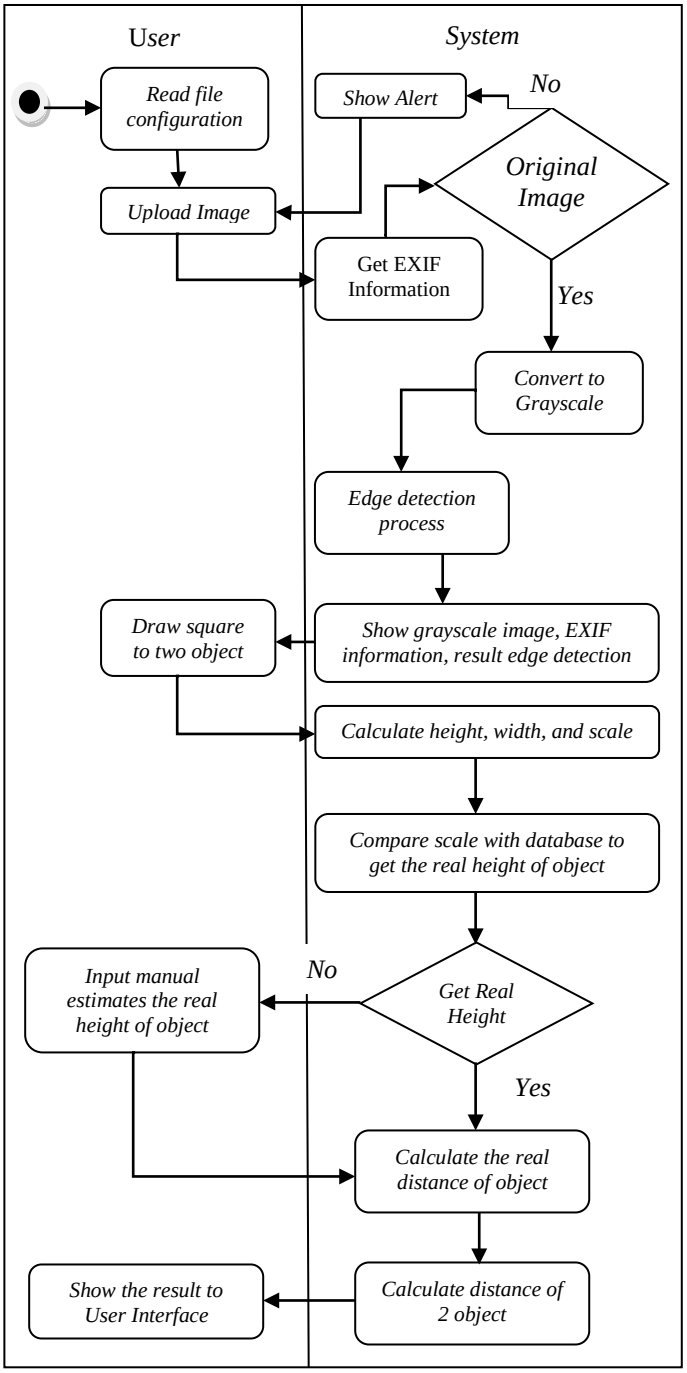
Gambar 4.*Use Case Diagram* Sistem

B. *Activity Diagram*

Activity diagram adalah representasi grafis dari seluruh tahapan alur kerja. *Activity diagram* digunakan untuk memodelkan perilaku di dalam suatu bisnis. *Activity diagram* dapat dilihat sebagai sebuah *sophisticated Data Flow Diagram* (DFD) yang digunakan pada analisis structural. Akan tetapi, berbeda dengan DFD, *activity diagram* mempunyai notasi untuk memodelkan aktivitas yang berlangsung secara paralel, bersamaan, dan juga proses pengambilan keputusan yang kompleks [14]. A*ctivity diagram* dari sistem yang dibuat diperlihatkan oleh Gambar 5.

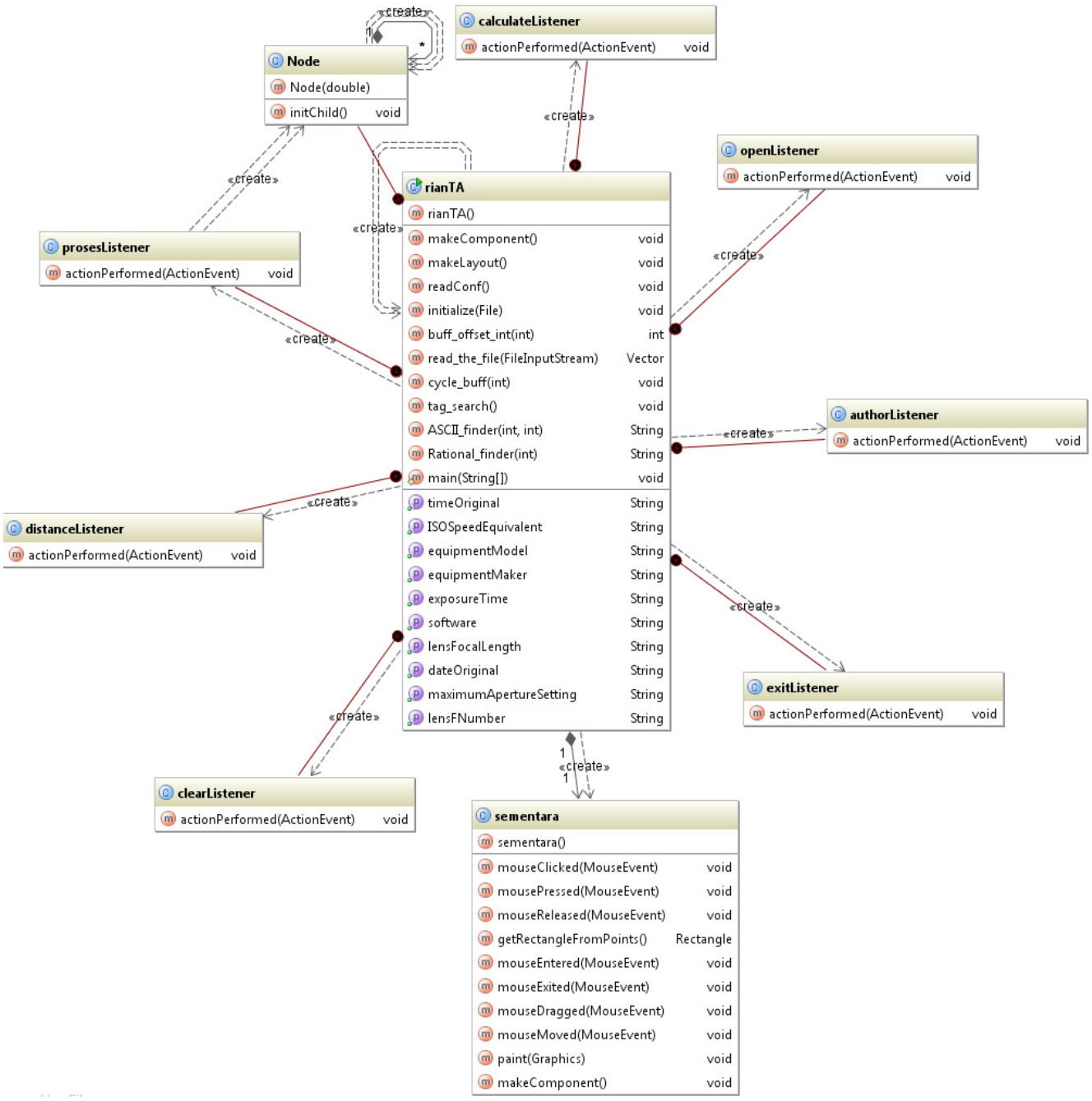
C. *Class Diagram*

Class diagram merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan dan tugas entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut dan operasi dari suatu *class* dan constraint yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi *class, relationships, associations, generalization, aggregation, attributes, methods/operations*, dan *visibility* [15]. Gambar 6 merupakan *class diagram* dari sistem ini.



Gambar 5. *Activity Diagram* Sistem.

.



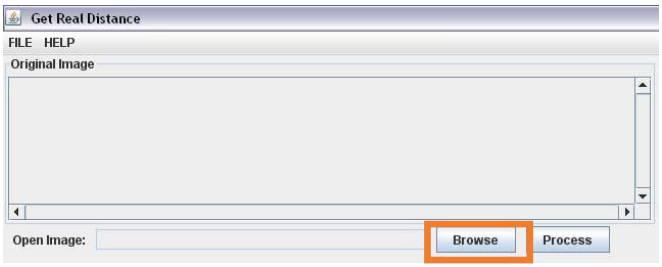
Gambar 6. *Class Diagram* Sistem.

D. Implementasi Sistem

Sistem diimplementasi dengan Java GUI *programming*. Pada sistem ini terdapat lima proses utama yaitu: input foto, ekstrak informasi EXIF, *convert grayscale*, proses deteksi tepi, dan perhitungan jarak. Berikut ini akan dijelaskan lebih detail mengenai proses-proses tersebut.

- Input Foto

  Saat sistem pertama kali dijalankan, *user* akan disajikan tampilan GUI seperti yang diperlihatkan pada Gambar 7. Untuk menginputkan sebuah foto (*image*), *user* dapat menekan tombol *browse* dan mencari foto yang diinginkan.

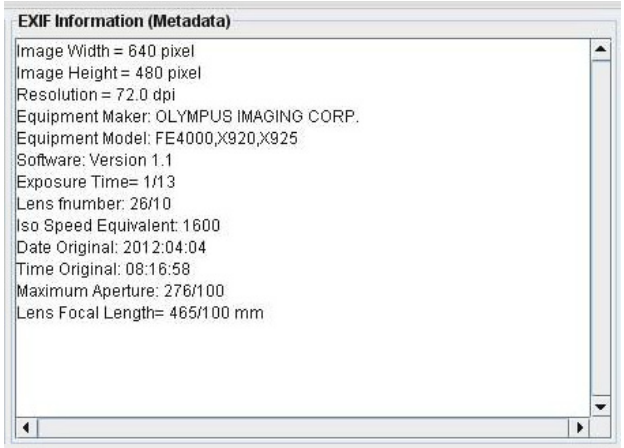


Gambar 7. GUI Sistem.

Setelah *user* menginputkan sebuah foto, sistem akan mengecek apakah foto tersebut original atau tidak. Untuk dapat mendeteksi apakah foto tersebut *original* atau tidak, maka akan dilakukan proses ekstraksi informasi EXIF pada foto tersebut.

- Ekstraksi Informasi EXIF

  Proses ekstraksi informasi EXIF pada suatu foto dilakukan dengan memanfaatkan library yang dibuat oleh Jeff Lynn dan Lane Schwartz (sumber: http://www.docjar.org/html/api/org/ydp/gfx/GraphicRead.java.html). Setelah informasi EXIF didapatkan, sistem akan mengecek apakah foto tersebut memiliki nilai *focal length* atau tidak. Apabila sebuah foto tidak memiliki nilai *focal length*, maka foto tersebut sudah mengalami proses editing sebelumnya (tidak *original*). *User* harus menginputkan foto yang lain, jika foto sebelumnya tidak original. Informasi EXIF akan ditampilkan ke *user* seperti yang diperlihatkan pada Gambar 8.



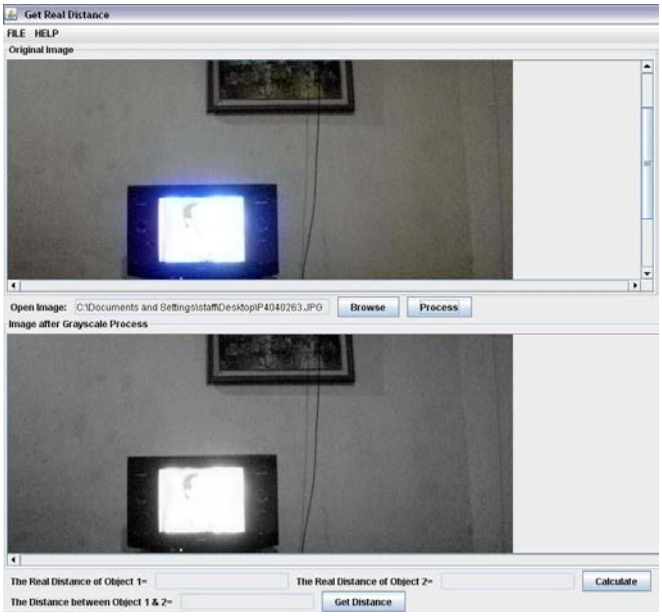
Gambar 8. Tampilan Informasi EXIF.

- *Convert Grayscale*

  Sebuah foto digital tersusun dari *pixel-pixel* warna yaitu *Red, Green, Blue* atau bisa disingkat RGB. Sistem akan melakukan ekstraksi nilai RGB untuk tiap *pixel* warna, yang nantinya akan diubah menjadi menjadi warna keabuan (*grayscale*). Warna keabuan ini diperoleh dengan menghitung rata-rata dari nilai RGB yang didapatkan tiap *pixel*, sesuai dengan persamaan (5).

$$Gray = \frac{Red+Green+Blue}{3} \quad (5)$$

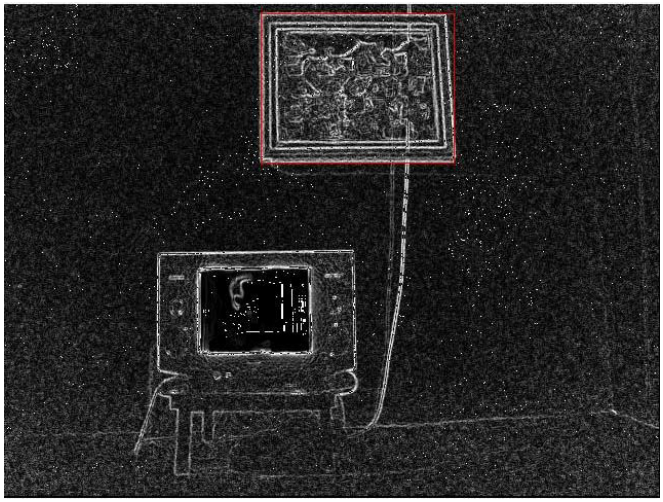
  Hasil perhitungan nilai *gray* tiap *pixel* akan disimpan pada sebuah *binary tree*. Tahapan inilah yang menjadi awal untuk proses deteksi tepi (*edge detection*). *Output* dari proses *convert grayscale* ini, akan ditampilkan di GUI yang disediakan seperti Gambar 9 berikut ini.



Gambar 9. *Output Convert Grayscale.*

- Proses Deteksi Tepi

  Proses selanjutnya yang akan dilakukan setelah *convert grayscale* adalah deteksi tepi. Proses ini akan menggunakan operator Sobel, seperti yang sudah dijelaskan sebelumnya. Gambar 10 memperlihatkan hasil dari proses deteksi tepi dengan operator Sobel.



Gambar 10. Hasil Deteksi Tepi.

- Perhitungan Jarak
  Setelah proses deteksi tepi dilakukan, *user* akan melakukan seleksi terhadap objek pada foto yang ingin dihitung jarak nyatanya. Selain itu, user juga harus menginputkan tinggi nyata dari objek yang diseleksi (dalam cm). Perhitungan jarak dilakukan dengan menggunakan persamaan (2) dan (3). Output akhir dari sistem adalah jarak nyata objek 1 dan 2 terhadap posisi kamera dan jarak sebenarnya dari objek 1 dan 2.

Gambar 11. Output Akhir

E. Pengujian Sistem

Pengujian dilakukan untuk mengetahui berapa tingkat akurasi dari sistem yang telah dibuat. Pengujian dilakukan terhadap barang-barang yang dapat ditemukan di kehidupan sehari-hari, seperti: botol minuman, gelas, tas ransel, televisi, lukisan, dan sebagainya. Tipe kamera yang digunakan adalah Nikon D3200. Toleransi maksimal selisih antara pengukuran manual yang dilakukan peneliti dengan perhitungan jarak oleh sistem adalah 10 cm. Tabel 1 menunjukkan hasil uji coba untuk perhitungan jarak nyata antara objek dengan posisi kamera yang telah dilakukan. Sedangkan Tabel 2 memperlihatkan hasil pengujian sistem untuk perhitungan jarak nyata antara objek 1 dan 2 pada foto.

Tabel 1. Uji Coba Perhitungan Jarak Objek Dengan Kamera.

Pengukuran/Perhitungan (cm)		Selisih	Toleransi
Manual	Sistem		
527	511	16	×
520	525	5	√
515	531	16	×
520	528	8	√
534	508	26	×
546	500	46	×
505	506	1	√
545	548	3	√
541	532	9	√
522	519	3	√
536	519	17	×
545	543	2	√
515	504	11	×
523	515	8	√
501	519	18	×
550	522	28	×
506	515	9	√
529	535	6	√
516	507	9	√
510	530	20	×
Total		√	**11**
		×	**9**
Akurasi Sistem (%)			**55%**

Tabel 2. Uji Coba Perhitungan Jarak Nyata 2 Objek

Pengukuran/Perhitungan (cm)		Selisih	Toleransi
Manual	Sistem		
140	125	15	×
146	145	1	√
126	135	9	√
103	100	3	√
115	118	3	√
128	120	8	√
145	106	39	×
109	123	14	×
125	108	17	×
127	122	5	√
Total		√	**6**
		×	**4**
Akurasi Sistem (%)			**60%**

IV. KESIMPULAN

Berdasarkan hasil uji coba yang telah dilakukan sebelumnya, maka dapat diambil kesimpulan bahwa akurasi sistem dalam menghitung jarak sebenarnya antara objek foto dengan posisi kamera adalah 55%. Selain itu untuk

perhitungan jarak nyata antara 2 objek pada suatu foto, sistem menghasilkan akurasi sebesar 60%.

Sistem ini masih membutuhkan banyak pengembangan lebih lanjut karena tingkat akurasi masih berkisar antara 55%-60%. Untuk pengembangan lebih lanjut dari sistem ini, diperlukan suatu metode untuk mendeteksi sudut pandang kamera dan untuk membedakan banyak benda (objek) pada foto secara otomatis.

REFERENSI

[1] Putri, A.R. (2016). Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya. *Jurnal Ilmiah Pendidikan Informatika*, Vol. 1, No. 1, Hal. 1-6.

[2] Royce, E., Timotius, I.K. & Setyawan, I. (2012). Sistem Pendeteksi Senyum Berdasarkan Metode Edge Detection, Histogram Equalization, Dan Nearest Neighbor. *Jurnal Ilmiah Elektroteknika*, Vol.11, No.1, Hal. 75-82.

[3] Putra, A.N., Basukesti, A. & Nugraheny, D. (2013). Penerapan Metode Sobel Untuk Pengukuran Tinggi Badan Menggunakan Webcam. *COMPILER*, Vol. 2, No. 1.

[4] Utami, R.Z., Suksmadana, I.M.B. & Kanata, B. (2015). Menentukan Luas Objek Citra Dengan Teknik Deteksi Tepi. *Dielektrika*, Vol. 2, No. 1, Hal.11-17.

[5] Wiyagi, R.O & Mustar, M.Y. (2015). Deteksi Jarak Objek Bercahaya Secara Real Time Menggunakan Kamera Tunggal. *Prosiding dari The 3rd Indonesian Symposium on Robot Soccer Competition 2015*.

[6] Kartowisastro, I.H. (2010). Pengukuran Jarak Berbasiskan Stereo Vision. *ComTech*, Vol. 1, No. 2, Hal. 598-605.

[7] Haryanti, F. (2017). Stereo Vision untuk Menentukan Jarak Objek dari Kamera Mobile Phone. (Skripsi). Program Studi Teknik Informatika, Universitas Dian Nuswantoro.

[8] Mulyawan, H., Samsono, M.Z.H. & Setiawardhana. (2011). *Identifikasi Dan Tracking Objek Berbasis Image Processing Secara Real Time*, Surabaya: Politeknik Elektronika Negeri Surabaya.

[9] Setiawan, R. & Bornok, M.B. (2015). *Estetika Fotografi*. Bandung: Universitas Katolik Parahyangan.

[10] Rosmala, D & Kresna, G. (2012). Implementasi Algoritma Binary Tree Pada Sistem Informasi Multilevel Marketing, *Jurnal Informatika*, Vol. 3, No. 3, Hal. 39-47.

[11] Sukatmi. (2017). Perbandingan Deteksi Tepi Citra Digital dengan Metode Prewitt, Sobel dan Canny. *Jurnal Ilmiah Manajemen Informatika dan Komputer*, Vol. 01, No. 01, Hal. 1-4.

[12] Wijaya, E. (2012). Analisis Intensitas Metode Pendeteksian Tepi Sobel. *Jurnal Komputer dan Informatika*, Edisi I, Vol. 1, Hal. 25-27.

[13] Hendini, A. (2016). Pemodelan Uml Sistem Informasi Monitoring Penjualan Dan Stok Barang (Studi Kasus: Distro Zhezha Pontianak). *Jurnal Khatulistiwa Informatika*, Vol. IV, No. 2, Hal. 107-116.

[14] Suryasari, Callista, A., & Sari, J. (2012). Rancangan Aplikasi Customer Service Pada PT. Lancar Makmur Bersama. *Jurnal Sistem Informasi*, Vol. 4, No. 2, Hal. 468-476.

[15] Urva, G & Siregar, H.F. (2015). Permodelan UML E-Marketing Minyak Goreng. *Jurnal Teknologi dan Sistem Informasi*, Vol. 1, No. 2, Hal. 92-101.