# BAB II
# LANDASAN TEORI

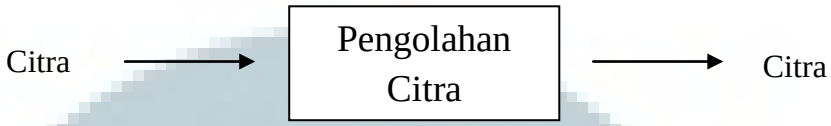
## 2.1 Pengertian Citra Digital

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsivisual (Putra, 2010). Citra yang dimaksud adalah citra digital untuk membedakannya dengan citra lain seperti foto dan lain-lain. Proses ini mempunyai data masukan dan informasi keluaran yang berbentuk citra. Hanya citra yang berbentuk digital yang dapat diproses oleh komputer digital. Data citra yang dimasukkan berupa nilai-nilai integer yang menunjukkan nilai intensitas cahaya atau tingkat keabuan setiap piksel.Citra digital dapat diperoleh secara otomatik dari sistem penangkap citra membentuk suatu matriks di mana elemen-elemennya menyatakan nilai intensitas cahaya pada suatu himpunan diskrit dari titik.

Setiap elemen dari array di atas disebut piksel yang merupakan suatu daerah empat persegi kecil dengan ukuran tertentu dan menunjukkan harga intensitas keabuan piksel padalokasi yang bersangkutan. Ukuran piksel ini sering disebut sebagai resolusi piksel.

Pengolahan citrabertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer). Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih

baik daripada citra masukan. Termasuk ke dalam bidang ini juga adalah pemampatan citra (*image compression*).




**Gambar 2.1 Pengolahan Citra**

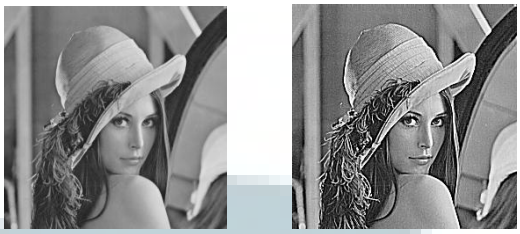
Komponen utama dari perangkat keras pengolahan citra secara digital adalah komputer dan alat peraga (Rinaldi,2004). Komputer tersebut bisa dari jenis komputer multi guna atau dari jenis khusus yang dirancang untuk pengolahan citra digital. Ada beberapa operasi yang dapat dilakukan oleh pengolahan citra antara lain:

1. Perbaikan kualitas citra (*image enhancement*).

Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan. Contoh-contoh operasi perbaikan citra:

a. perbaikan kontras gelap/terang
b. perbaikan tepian objek (*edge enhancement*)
c. penajaman (*sharpening*)
d. pemberian warna semu (*pseudocoloring*)
e. penapisan derau (*noise filtering*)

Gambar 2.2 adalah contoh operasi penajaman. Operasi ini menerima masukan sebuah citra yang gambarnya hendak dibuat tampak lebih tajam. Bagian citra yang ditajamkan adalah tepi-tepi objek.



(a) (b)

**Gambar2.2 (a) Citra Lena asli, (b) Citra Lena setelah ditajamkan**

2. Pemugaran citra (*image restoration*).

Operasi ini bertujuan menghilangkan/meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui. Contoh-contoh operasi pemugaran citra:

a. penghilangan kesamaran (*deblurring*)

b. penghilangan derau (*noise*)

3. Pemampatan citra (*image compression*).

Jenis operasi ini dilakukan agar citra dapat direpresentasikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai kualitas gambar yang bagus. Contoh metode pemampatan citra adalah metode JPEG. Perhatikan Gambar 2.3. Gambar sebelah kiri adalah citra kapal yang berukuran 258 KB. Hasil pemampatan citra dengan metode JPEG dapat mereduksi ukuran citra semula sehingga menjadi 49 KB saja.

(a)

(b)

**Gambar 2.3(a) Citra boat.bmp (258 KB) sebelum dimampatkan, (b) citra boat.jpg (49 KB) sesudah dimampatkan.**

4. Pengorakan citra (*image analysis*)

Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi kadang kala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh-contoh operasi pengorakan citra:

a. Pendeteksian tepi objek (*edge detection*)

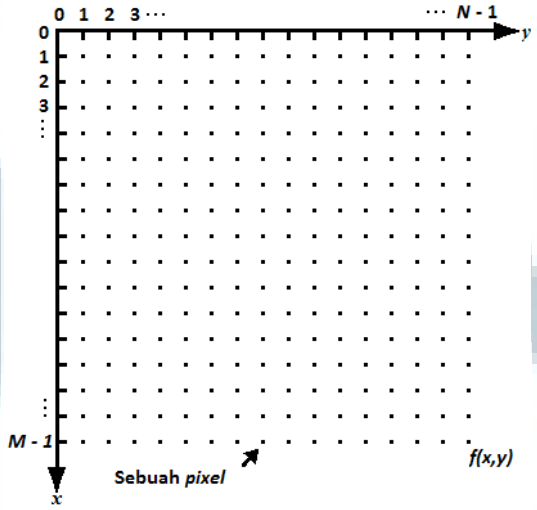
b. Ekstraksi batas (*boundary*)

c. Representasi daerah (*region*)

5. Rekonstruksi citra (*image reconstruction*)

Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekonstruksi citra banyak digunakan dalam bidang medis. Misalnya beberapa foto *rontgen* dengan sinar Xdigunakan untuk membentuk ulang gambar organ tubuh.

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data dua dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit tertentu. Suatu citra dapat didefinisikan sebagai fungsi *f(x,y)* berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial. Apabila nilai x, y dan nilai amplitudo f secara keseluruhan berhingga (*finite*) dan bernilai diskrit maka dikatakan bahwa citra tersebut adalah citra digital. Gambar 2.4 menunjukkan posisi koordinat citra digital.



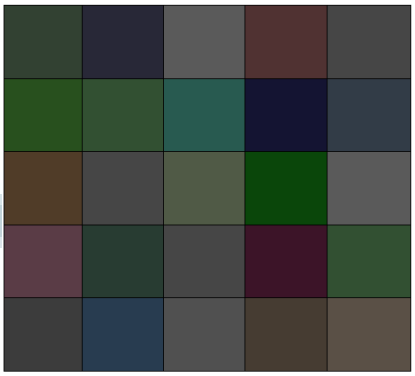

**Gambar 2.4Koordinat Citra Digital**

## 2.2 Jenis Citra Digital

Ada banyak cara untuk menyimpan citra digital di dalam memori. Cara penyimpanan menentukan jenis citra digital yang terbentuk. Beberapa jenis citra digital yang sering digunakan adalah citra warna.

1. Citra Warna (*True Color*)

   Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (RGB = *Red Green Blue*). Setiap warna dasar merupakan penyimpanan 8 bit = 1*byte*, yang berarti setiap warna mempunyai gradasi sebanyak 255 warna. Berarti setiap piksel mempunyai kombinasi warna sebanyak $2^8.2^8.2^8. = 2^{24}$ = 16 juta warna lebih. Itulah sebabnya format ini dinamakan *true color* karena mempunyai jumlah warna yang cukup besar sehingga dikatakan hampir mencakup semua warna di alam. Pada *true color* 1 piksel citra diwakili oleh 3 *byte*, dimana masing-masing *byte* merepresentasikan warna merah (*Red*), hijau (*Green*), dan biru (*Blue*). Gambar 2.5 adalah contoh penyimpanan citra warna di dalam memori.




**Penyimpanan di dalam Memori**

| | | | |
|---|---|---|---|
| R = 50<br>G = 65<br>B = 50 | R = 40<br>G = 40<br>B = 55 | R = 80<br>G = 50<br>B = 50 | R = 70<br>G = 70<br>B = 70 |
| R = 40<br>G = 80<br>B = 30 | R = 50<br>G = 80<br>B = 50 | R = 20<br>G = 20<br>B = 50 | R = 50<br>G = 60<br>B = 70 |
| R = 80<br>G = 60<br>B = 40 | R = 70<br>G = 70<br>B = 70 | R = 80<br>G = 90<br>B = 70 | R = 90<br>G = 90<br>B = 90 |
| R = 90<br>G = 60<br>B = 70 | R = 40<br>G = 60<br>B = 50 | R = 70<br>G = 70<br>B = 70 | R = 50<br>G = 80<br>B = 50 |
| R = 60<br>G = 60<br>B = 60 | R = 40<br>G = 60<br>B = 80 | R = 80<br>G = 80<br>B = 80 | R = 90<br>G = 80<br>B = 70 |

**Gambar 2.5Contoh Penyimpanan Citra Warna di dalam Memori**

## 2.3 Background Modeling

*Background Modeling* sering digunakan untuk mendeteksi objek bergerak dari kamera statis. *Background Modeling* juga digunakan pada aplikasi yang berbeda untuk model latar belakang dan kemudian mendeteksi objek bergerak dalam adegan seperti pada video pengawas, optik penangkap gerakan, dan multimedia (Bouwmans, 2015). Untuk mendapatkan latar belakang (*background*) dari sebuah video dapat dicari dengan rumus:

$$B(x,y,t) = \frac{1}{n}\sum_{i=0}^{n-1} I(x,y,t-i)$$

Dari rumus di atas, *B*(*x*,*y*,*t*) merupakan variabel latar belakang pada dengan titik koordinat *x* dan *y*, pada waktu *t*. Sedangkan *I*(*x*,*y*,*t*) merupakan variabel untuk citra (*frame* video). Variabel latar belakang di dapat dari nilai rata-rata dari variabel citra (*frame* video) pada waktu sekarang sampai *n*. Contoh kasusnya adalah ketika merekam selama 2 detik dan diambil 25 frame setiap satu detiknya. Setiap frame/image memiliki panjang dan lebar dengan satuan pixel. Setiap pixelnya terdiri dari nilai warna merah (R), hijau (G), dan biru (B).

**Tabel 2.1Perhitungan Pixel Pertama**

| | |
|---|---|
| (20,123,148) | (255,1,200) |
| (30,31,32) | (230,27,36) |

**Tabel2.2 Perhitungan Pixel Kedua**

| | |
|---|---|
| (180,227,230) | (222,225,222) |
| (241,177,120) | (123,45,67) |

Tabel di atas adalah nilai dari setiap frame dengan ukuran 2x2 pixel. Berikut ini adalah perhitungannya:

**Tabel 2.3 Hasil Perhitungan Pixel Pertama dan Kedua**

| | |
|---|---|
| (20+180) /2,<br>(123+227)/2,<br>(148+230)/2 | (255+222)/2,<br>(1+225)/2,<br>(200+222)/2 |
| (241+30)/2,<br>(31+177)/2,<br>(32+120)/2 | (230+123)/2,<br>(27+45)/2,<br>(36+67)/2 |

Cara perhitungan pada tabel 2.3 adalah dengan menghitung masing-masing dari nilai R(Red), G(Green), B(Blue). Maka, hasil yang didapat adalah Angka pertama *Red* dari tabel pertama yaitu 20 ditambahkan dengan angka kedua *Red* dari tabel kedua yaitu 180 kemudian ketika sudah dijumlahkan, hasilnya dibagi dengan 2 sehingga didapatkanlah hasil dari nilai *Red* pada detik pertama yang berjumlah 100.

## 2.4 Pengertian Video Digital

Kata video berasal dari kata Latin, yang berarti “saya lihat”. Video adalah teknologi pemrosesan sinyal elektronik yang mewakilkan gambar bergerak. Aplikasi umum dari teknologi video adalah televisi. Video juga dapat digunakan dalam aplikasi teknik, keilmuan, produksi, dan keamanan.

Video digital dapat disebut array 3 dimensi dari piksel berwarna. 2 dimensi melayani arah spasial dari gambar bergerak (horizontal dan vertikal) dan satu dimensi lainnya akan merepresentasikan domain waktu.

## 2.5 Audio Video Interleaved (.avi)

Arsitektur video digital tersusun atas sebuah format untuk mengodekan dan memainkan kembali file video dengan komputer dan menyertakan sebuah pemutar (*player*) yang mengenali dan membuka file yang dibuat untuk format tersebut. Contohnya arsitektur video digital Microsoft Windows Media Format, format video tersebut adalah Audio Video Interleaved (.avi).

## 2.6 Deteksi Gerak (*Motion Detection*)

Deteksi Gerak adalah proses mendeteksi perubahan posisi suatu objek relatif terhadap lingkungannya atau perubahan lingkungan relatif terhadap suatu objek. Proses deteksi gerak disini adalah memanfaatkan video dengan

menganalisa frame video kemudian menentukan apakah terjadi pergerakan pada video tersebut.

## 2.7 Algoritma Simple Background Modeling Motion Detector

Algoritma *Simple Background Modeling Motion Detector* adalah algoritma deteksi yang menemukan gerakan dengan didasarkan pada perbedaan antara frame video dengan frame yang menjadi latar belakang (*background*). Dengan fitur *background modeling* ini dapat memberikan kemampuan untuk menyoroti lebih tepat daerah gerak.


**Gambar 2.6Background Modeling Motion Detector**

## 2.8 Blob Detection

Bagi banyak pengolahan citra, mendeteksi objek *low-level* dalam sebuah gambar merupakan hal yang sangat penting. Objek dalam bentuk duaatau tiga dimensi tersebut biasa disebut *blob.* Bentuk *blob* timbul dalam cara yang berbeda bergantung pada ukuran dan dapat di deteksi dengan menggunakan metode sederhana dalam sebuah representasi gambar.

Ada beberapa teori mengenai definisi *blob* yang dikemukakan oleh beberapa orang, yaitu antara lain:

1. Lindeberg mendefinisikan sebuah *blob* sebagai sebuah daerah yang terkait setidaknya sau daerah yang extreme, baik maksimum maupun minimum atau sebuah daerah etrang maupun daerah gelap.
2. Hinz mendefinisikan sebuah *blob* sebagai sebuah bujur sangkar dengan daerah yang homogen.
3. Rosenfeld dan Sher mendefinsiikan sebuah *blob* sebagai sebuah persimpangan dari garis tegak lurus ke daerah tepi dari tangennya, dikelilingi oleh enam arah atau lebih.
4. Yang dan Parvin mendefinsiikan sebuah *blob 3D* sebagai fitur berbentuk bulat dalam sebuah skala ruang.

Untuk berbagai macam tipe *blob,* berbagai metode berbeda dibutuhkan. Namun metode-metode tersebut harus memenuhi beberapa syarat:

1. Handal dan dapat dipercaya. Metode pada citra *low-level* harus tahan terhadap *noise*.

2. Akurasi. Dalam metrologi citra dengan hasil akurasi tinggi sangat diperlukan.
3. Skalabilitas. Blob dalam ukuran yang berbeda harus dapat terdeteksi.
4. Kecepatan. Metode tersebut harus mendekati proses *real-time*.
5. Mempunyai parameter yang sedikit dan semantic. Metode tersebut harus mudah dimengerti agar mudah disesuaikan.
6. Mampu mengekstraksi ciri secara geometris dan radiometric. Hal ini dibutuhkan untuk klasifikasi *blob.*

*Blob Detection* merupakan sebuah algoritma yang digunakan untuk menentukan apakah sekelompok menghubungkan piksel yang terkait satu sama lain. Hal ini berguna untuk mengidentifikasi objek terpisah dalam suatu *scene*, atau menghitung jumlah objek dalam suatu *scene*.

Sebagian besar metode *Blob Detection* didasarkan pada representasi skala ruang. Tujuan utama dari representasi skala ruang adalah unutk memahami struktur gambar pada semua tingkat resolusi secara bersamaan dan gambar dalam berbagai skala. Skala ruang diperoleh dengan menerapkan *smoothing kernel* seperti Gaussian, pada gambar dengan parameter skala tergantung pada jumlah smoothing yang dibutuhkan. Beberapa metode yang dapat dipakai antaranya:

1. *Template matching*. Sebuah template akan dipindahkan ke atas gambar untuk dicari dan blob akan terdeteksi dimana template tersebut cocok dengan bagian dari gambar.
2. *Watershed detection*. Metode ini mengasumsikan sebuah citra menjadi tumpukan nilai abu-abu dan menyimulasikan proses hujan yang jatuh ke

tumpukan tersebut, mengalir di tumpukan tersebut dan terakumulasi pada cekungan.

3. *Spoke filter*. Metode yang ditemukan oleh Minor dan Sklansky ini dapat mendeteksi *blob* dalam berbagai ukuran dengan menggunakan sebuah filter yang dinamakan *spoke filter* atau biasa juga disebut Adaptive Spatial Erosion Filter.
4. *Automatic scale selection.* Prinsip dari metode ini adalah mengasumsikan sebuah scale level, dimana beberapa kombinasi dari turunan normalisasinya mengasumsikan sebuah nilai maksimum atas scale tersebut, yang mencerminkan ukuran *blob* yang sesuai. Metode ini dikemukakan oleh Lindeberg.
5. *Sub-pixel pricise blob detection.* Metode ini mendefinisikan *blob* sebagai sebuah bujur sangkar dengan kontras yang konstan, yang menjadi titik ekstrem dibawah Gaussian *smoothing.* Metode ini dikemukakan oleh Hinz.
6. *Effective maxima line detection.* Darmeval menyajikan sebuah metode di mana kurva modulus dalam skala yang berbeda, yang disebut *maxima lines*, dipilih secara efektif untuk memasukkan *blob* dari *noise.*
7. *Confidence measurement.* Forssen dan Grandlund mengemukakan sebuah metode yang kompleks untuk mengektraksi *blob* dari sebuah gambar tanpa menggunakan*Gaussiansmoothing*. Metode ini akan membagi channel citra dengan menggunakan fungsi kernel dan menghasilkan sebuah *low-pass pyramid* serta menggunakan sebuah *threshold.*

## 2.9 Algoritma Blob Counting Objects Processing

Algoritma *Blob Counting Objects Processing* adalah salah satu algoritma pemrosesan gerakan. Algoritma ini memungkinkan untuk menghitung objek yang terpisah pada frame *motion* yang sudah diproses oleh algoritma deteksi. Di samping itu dengan algoritma ini dapat menyoroti objek terdeteksi dengan persegi panjang. Objek yang diproses pada algoritma ini adalah objek yang bergerak dengan ukuran yang ditentukan dan dapat menolak objek yang lebih kecil dari ukuran yang telah ditentukan. Algoritma ini seharusnya hanya digunakan dengan algoritma deteksi gerak yang mungkin akurat menemukan benda bergerak.


**Gambar 2.7Blob Counting Objects Processing**

## 2.10 *Gaussian Filter*

*Gaussian Filter* digunakan untuk meminimalisir *noise* yang ada pada gambar sehingga gambar terlihat lebih halus daripada gambar aslinya. *Gaussian Filter* sendiri terdiri dari 2 bagian yaitu:

1. *Gaussian Lowpass Filters*

   *Gaussian Lowpass Filters* (GLPF) bisa dimanfaatkan untuk mengetahui lebih dalam relasi antara *spatial* dengan *frequency domain.*

2. *Gaussian Highpass Filters*

   Tampilan *Gaussian Highpass Filters* (GHPF) yang dihasilkan akan nampak lebih lembut, meskipun *filtering* dari objek-objek yang lebih kecil nampak lebih bersih dengan *Gaussian filter*.

Ini adalah contoh potongan *code* dari *Gaussian Filter* yang digunakan untuk memberi efek sedikit *blur* pada gambar agar terlihat lebih halus.

```
private void monitor_NewFrame(object sender, ref Bitmap image)
{
    lock (this)
    {
        GaussianBlur(image, 1);
        pendeteksi.ProcessFrame(image);
        BlobCountingObjectsProcessing penghitungDeteksi =
        (BlobCountingObjectsProcessing)pendeteksi.MotionProcessingAlgorithm;
        hitungObjekTerdeteksi = penghitungDeteksi.ObjectsCount;

        if (rekamBuka)
        {
            saveFrame(image);
            if (hitungObjekTerdeteksi > 0) prosesRekamVideo(image);
        }
    }
}
```

**Gambar 2.8*Code Gaussian Filter***

## 2.11 *Grayscale*

Grayscale atau abu-abu pada sebuah image digital adalah image yang pada setiap pixelnya hanya berisikan informasi intensitas warna putih dan hitam. Image Grayscale memiliki banyak variasi nuansa abu-abu sehingga berbeda dengan image hitam-putih.

Grayscale juga disebut monokromatik karna tidak memiliki warna lain selain variasi intensitas putih dan hitam. Sebuah image yang dijadikan Grayscale akan terkesan berbeda bila dibandingkan dengan image berwarna.

Untuk mengubah warna objek menjadi abu-abu, dapat menggunakan *code* yang dapat dilihat pada gambar 2.9

```
private void monitor_NewFrame(object sender, ref Bitmap image)
{
    lock (this)
    {
        GrayScaleEdit(image);
        pendeteksi.ProcessFrame(image);
        BlobCountingObjectsProcessing penghitungDeteksi =
        (BlobCountingObjectsProcessing)pendeteksi.MotionProcessingAlgorithm;
        hitungObjekTerdeteksi = penghitungDeteksi.ObjectsCount;

        if (rekamBuka)
        {
            saveFrame(image);
            if (hitungObjekTerdeteksi > 0) prosesRekamVideo(image);
        }
    }
}
```

**Gambar 2.9*Code Grayscale***

## 2.12 *Threshold*

*Threshold* adalah besaran numerik pada *output*yang berhubungan degan perubahan *input* (Mehmet, 2004). Tujuan dari *threshold* adalah untuk mengekstrak piksel dari beberapa gambar yang mewakili suatu objek.

*Threshold*juga digunakan sebagai sensitivitas antara nilai yang 1 dengan nilai yang lainnya. Semakin kecil nilai *threshold*nya, maka akan semakin sensitif dan akurat nilai objek yang direkam.

## 2.13 AForge.NET Framework

Aforge.NET adalah *framework* C# berbasis *opensource* yang dirancang dan dikembangkan untuk para peneliti di bidang komputer visi, kecerdasan buatan, pengolahan citra, jaringan syaraf tiruan, algoritma genetika, logika fuzzy, dan lainnya.

*Framework* ini terdiri dari kumpulan *library* dan contoh kode program yang dibagi berdasarkan karakteristiknya sebagai berikut:

1. *AForge.Imaging*, merupakan *library* tentang pengolahan citra.
2. *AForge.Vision*, merupakan *library* tentang komputer visi.
3. *AForge.Video*, merupakan *library* tentang pengolahan video.
4. *AForge.Neuro*, merupakan *library* tentang jaringan syaraf tiruan.
5. *AForge.Genetic*, merupakan *library* tentang algoritma genetika.
6. *AForge.Fuzzy*, merupakan *library* tentang algoritma *fuzzy*.
7. *AForge.Robotics*, merupakan *library* tentang bidang robotika.