BAB II
LANDASAN TEORI

2.1 Tinjauan Studi

Pada penelitian yang dilakukan ini menggunakan berbagai macam referensi yang berasal dari penelitian-penelitian sebelumnya. Penggunaan referensi tersebut digunakan untuk proses pengerjaan penelitian dengan menggunakan teknologi dalam pengolahan citra yang sesuai dengan perkembangan keilmuan untuk mempertimbangkan metode yang akan digunakan dalam penelitian ini. Berikut merupakan penelitian yang memaparkan tentang penyakit *infectious myonecrosis virus* (IMNV).

Menurut penelitian yang pernah dilakukan oleh Kathy (2019) dengan judul "Panduan strategi IMNV udang", IMNV merupakan penyakit virus yang menyebabkan kematian substansial pada populasi budidaya udang vaname. IMNV tersebut menginfeksi ke area nekrotik putih yang luas di otot lurik, terutama di segmen distal abdomen dan ekor kipas. Untuk dapat mendiagnosis penyakit tersebut, biasanya dilakukan dengan melihat tanda klinis kasar secara visual yang terdapat pada udang. Diagnosis tersebut biasanya terlalu lambat untuk memungkinkan keputusan manajemen praktis oleh petambak udang. Dengan alasan tersebut metode diagnostik molekuler dengan *polymerase chain reaction* (PCR) dapat digunakan untuk mendeteksi virus lebih cepat [8].

Dari hasil penelitian yang dilakukan oleh Humidah (2018) bahwasanya hingga saat ini, metode pengobatan infeksi oleh virus tersebut belum ditemukan sehingga usaha yang dapat dilakukan yaitu dengan pencegahan. Salah satu metode pencegahan yaitu pengetahuan mengenai status penyebaran IMNV. Selain itu, cara lain untuk dapat mencegahnya yaitu dengan pendekteksian virus menggunakan metode PCR dan analisa histopatologi.

Dari penelitian yang sudah dilakukan oleh Kathy dan Humidah diatas, IMNV merupakan penyakit utama yang terdapat pada budidaya udang vaname yang dapat menyebabkan kerugian bagi petambak yang membudidayakannya. Hingga saat ini

disebutkan belum terdapatnya metode pengobatan oleh virus tersebut. Usaha yang dapat dilakukan untuk menindaklanjuti penyebaran virus tersebut yaitu dengan pencegahan yang salah satunya dengan metode *polymerase chain reaction* (PCR) dan analisa histopatologi. Dikarenakan harganya yang mahal, pengujian menggunakan PCR hanya bisa dilakukan di tempat-tempat tertentu, seperti Balai Pengembangan Budidaya Air Payau (BBPBAP) [12].

Oleh sebab itu, dalam penelitian mencari alternatif baru yaitu dengan memanfaatkan teknologi informasi menggunakan pengolahan citra digital dengan program komputer yang dapat mengenali karakteristik gejala penyakit udang yang terdapat pada udang vaname secara visual. Dengan menggunakan pengolahan citra digital dapat memungkinkan digunakan untuk membantu mendeteksi penyakit IMNV pada udang. Ada beberapa cara agar dapat menggunakan pemrosesan citra digital untuk membedakan udang normal (sehat) dari udang yang terinfeksi dengan gejala IMNV dalam berupa perbedaan warna putih kapas yang terdapat pada tubuh udang secara tekstur dan warna serta metode yang dapat membedakan udang tersebut dari hasil ekstraksi. Adapun literatur penelitian sebelumnya yang digunakan sebagai rujukan penelitian ini adalah sebagai berikut.

Pada penelitian yang pernah dilakukan oleh Aarthy (2019) yaitu Klasifikasi kanker payudara berdasarkan termal gambar menggunakan support vector machine, teknik *grey level co-occurence matrix* (GLCM) digunakan untuk ekstraksi fitur dan *support vector machine* (SVM) untuk mengklasifikasikan input sebagai kanker atau non-kanker. Akurasi klasifikasi menunjukkan 97,6% yang secara signifikan baik dari percobaan 83 gambar yang terdiri dari 34 normal dan 49 abnormal (tumor ganas dan jinak) [13].

Penelitian sebelumnya pernah dilakukan oleh justiawan (2019) yaitu "analisis perbandingan sistem pencocokan warna untuk pengenalan gigi menggunakan *color moment*" dengan *color moment* sebagai ekstraksi warna mendapatkan akurasi terbesar 97.5% [14].

Penelitian pernah dilakukan oleh Santhi (2019) yaitu tentang "klasifikasi gambar resonansi magnetik menggunakan delapan arah *gray level co-occurrence matrix* (8dglcm) berdasarkan ekstraksi fitur" menggunakan metode GLCM untuk

mengekstraksi fitur tekstur dari suatu gambar. Hasil akhir menunjukkan kinerja sistem yang diusulkan adalah 95,65% [15].

Penelitian lain juga pernah dilakukan oleh Ahsani (2019) yaitu "temu kembali citra makanan menggunakan ekstraksi fitur *gray level co-occurrence matrix* dan CIE *L*a*b* color moments* untuk pencarian resep masakan" menggunakan GLCM sebagai metode ekstraksi fitur tekstur dan *color moment* sebagai metode ekstraksi warna yang dilakukan. Dari 31 data uji yang diberikan, menghasilkan akurasi sebesar 97,604% [16].

Penelitian yang pernah dilakukan oleh Khan (2019) yaitu "deteksi salju menggunakan kamera video dalam kendaraan dengan fitur gambar berbasis tekstur menggunakan *k-nearest neighbor*, *support vector machine*, dan *random fores*t" menggunakan tiga algoritma klasifikasi: *support vector machine* (SVM), *k-nearest neighbor* (K-NN), dan *random forest* (RF) yang digunakan. Akurasi prediksi keseluruhan tertinggi didapatkan dari algoritma klasifikasi SVM dengan ekstraksi tekstur berbeda yaitu 96% [17].

Penelitian lain pernah dilakukan oleh Asmara (2019) yaitu ekstraksi fitur iris menggunakan *gray level co-occurence matrix* dan *gabor kernel* menyaring dampaknya pada gambar kompresi iris *huffman*. Hasil uji fitur GLCM mencapai akurasi klasifikasi SVM 95,24%, sedangkan menggunakan *naive bayes* mencapai 85,71. Proses klasifikasi berdasarkan fitur GLCM menunjukkan bahwa metode SVM harus memiliki akurasi pengenalan tertinggi dibandingkan dengan *naive bayes classifier* [18].

Penelitian pernah dilakukan oleh Latha (2019) yaitu deteksi *septoria* pada gambar daun *blueberry* menggunakan SVM *classifier*. Penelitian tersebut melakukan ekstraksi fitur pada daun yang tersegmentasi dengan *gray level co-occurrence matrix* (GLCM) dan *support vector machine* (SVM) classifier untuk mengklasifikasikan daun yang terinfeksi atau tidak. Akurasi rata-rata mendeteksi *septoria* pada daun *blueberry* adalah 96,77% [19].

Penelitian lain pernah dilakukan oleh Devi (2019) yaitu "deteksi penyakit padi yang diaktifkan melalui web menggunakan penginderaan terkompresi" menggunakan

metode *support vector machine classifier* untuk mengklasifikasikan penyakit. Sistem yang diusulkan mencapai tingkat pengenalan penyakit 98,38% [20].

Penelitian pernah dilakukan oleh Shefali (2019) yaitu ”pendekatan kompeten dan baru dalam merancang sistem pengambilan gambar cerdas” menggunakan *color moment* mendapatkan akurasi terbesar yaitu 95.8% dalam mengenali gambar [22].

Penelitian lainnya yang dilakukan oleh Bahia (2019) yaitu ekstraksi fitur hibrida dan pendekatan *machine learning* untuk klasifikasi buah dan sayuran dengan identifikasi fitur tekstur menggunakan *histogram of oriented gradient*, *local binary pattern* dan *gray level co-occurrence matrix*. Hasil yang didapatkan menunjukkan bahwa akurasi SVM lebih tinggi dengan fungsi kernel kuadrat yaitu 94,3%, dibandingkan dengan akurasi KNN fungsi kernel *fine* yaitu 75.6% [23].

Pada penelitian yang pernah dilakukan oleh Fibrianda (2018) yaitu “analisis perbandingan akurasi deteksi serangan pada jaringan komputer dengan metode *naïve bayes* dan *support vector machine* (SVM)”. Akurasi prediksi keseluruhan tertinggi didapatkan dari algoritma klasifikasi *support vector machine* (SVM) dengan kernel *polynomial* yaitu sebesar 99.999% [24].

Penelitian pernah dilakukan oleh Rivan (2020) yaitu “pengenalan iris menggunakan fitur *local binary pattern* (LBP) dan *radial basis function* (RBF) *classifier*” menggunakan *local binary pattern* sebagai ekstraksi ciri mendapatkan akurasi sebesar 83,33% [25].

Berdasarkan hasil pemaparan penelitian yang pernah dilakukan oleh peneliti sebelumnya, metode ekstraksi *gray level co-occurence matrix (*GLCM) sebagai ekstraksi tekstur dan *color moment* sebagai ekstraksi warna pada gambar menghasilkan akurasi penelitian yang cukup baik diantara metode ekstraksi tekstur dan ekstraksi warna lainnya untuk mengekstraksi tektstur dan warna pada citra digital. Begitu juga dengan *support vector machine* (SVM) sebagai metode klasifikasi memiliki tingkat akurasi yang cukup baik dibandingkan dengan metode klasifikasi lainnya.

Untuk itu, pada penelitian ini akan menggunakan *gray level co-occurence matrix (*GLCM) sebagai ekstraksi tekstur dan *color moment* sebagai ekstraksi warna pada gambar untuk mengekstraksi bagian penyakit *infectious myonecrosis virus* (IMNV)

dan *support vector machine* (SVM) sebagai metode klasifikasi untuk melakukan deteksi penyakit *infectious myonecrosis virus* (IMNV) pada citra udang vaname.

Adapun rangkuman dari penelitian-penelitian terdahulu yang pernah dilakukan oleh peneliti dapat dilihat berupa metode yang digunakan dan hasil tingkat akurasi yang didapatkan pada tabel 2.1 berikut.

Tabel 2. 1 Rangkuman penelitian terkait

No.	Peneliti	Judul	Metode digunakan	Hasil
1.	Aarthy (2019)	Klasifikasi kanker payudara berdasarkan termal gambar menggunakan support vector machine, teknik *gray level co-occurence matrix* (GLCM) digunakan untuk ekstraksi fitur dan *support vector machine* (SVM) untuk mengklasifikasikan input sebagai kanker atau non-kanker [13]	*gray level co-occurence matrix* (GLCM)	Akurasi : 97,6%
2.	Justiawan (2019)	Analisis perbandingan sistem pencocokan warna untuk pengenalan gigi menggunakan *color moment* [14]	*color moment*	Akurasi : 97,5%
3.	Santhi (2019)	Klasifikasi gambar resonansi magnetik menggunakan delapan arah *gray level co-occurrence matrix* (8dglcm) berdasarkan ekstraksi fitur [15]	*gray level co-occurence matrix* (GLCM)	Akurasi : 95,65%
4.	Ahsani (2019)	Temu kembali citra makanan menggunakan ekstraksi fitur *gray level co-occurrence matrix* dan CIE *L*a*b* color moments* untuk pencarian resep masakan [16]	*gray level co-occurence matrix* (GLCM) dan *color moment*	Akurasi : 97,604%
5.	Khan (2019)	Deteksi salju menggunakan kamera video dalam kendaraan dengan fitur gambar berbasis tekstur menggunakan *k-nearest neighbor*, *support vector machine*, dan *random fores*t [17]	*support vector machine* (SVM)	Akurasi : 96%
6.	Asmara (2019)	Ekstraksi fitur iris menggunakan *gray level co-occurence matrix* dan *gabor kernel* menyaring dampaknya pada gambar kompresi iris *Huffman* [18]	*gray level co-occurrence matrix* (GLCM) dan *support vector machine* (SVM)	Akurasi : 95,24%
7.	Latha (2019)	Deteksi *septoria* pada gambar daun *blueberry* menggunakan SVM *classifier* [19]	*gray level co-occurrence matrix* (GLCM) dan *support vector machine* (SVM)	Akurasi : 96,77%

No.	Peneliti	Judul	Metode digunakan	Hasil
8.	Devi (2019)	Deteksi penyakit padi yang diaktifkan melalui web menggunakan penginderaan terkompresi [20]	*support vector machine* (SVM)	Akurasi : 98,38%
9.	Shefali (2019)	Pendekatan kompeten dan baru dalam merancang sistem pengambilan gambar cerdas [22]	*color moment*	Akurasi : 95.8%
10.	Bahia (2019)	Ekstraksi fitur hibrida dan pendekatan *machine learning* untuk klasifikasi buah dan sayuran dengan identifikasi fitur tekstur menggunakan *histogram of oriented gradient* (HOG), *local binary pattern* (LBP) dan *gray level co-occurrence matrix* (GLCM) [23]	*support vector machine* (SVM)	Akurasi : 94,3%
11.	Fibrianda (2018)	Analisis perbandingan akurasi deteksi serangan pada jaringan komputer dengan metode *naïve bayes* dan *support vector machine* (SVM) [24]	*support vector machine* (SVM)	Akurasi : 99.999%
12.	Rivan (2020)	Pengenalan iris menggunakan fitur *local binary pattern* (LBP) dan RBF *classifier* [25]	*local binary pattern* (LBP)	Akurasi : 83,33%

2.2 Tinjauan Pustaka

2.2.1 Udang vannamei (*Litopenaeus vannamei*)

Udang vaname atau lebih biasa disebut dengan udang vannamei (*Litopenaeus vannamei*) merupakan udang introduksi [1]. Udang introduksi merupakan jenis udang yang bukan habitat asli Indonesia, melainkan berasal dari perairan Amerika Tengah. Dampak dari kedatangan udang vaname turut berperan besar dalam meningkatnya jumlah produksi udang di Indonesia [2]. Adapun udang vaname dapat dilihat pada gambar 2.1.

Gambar 2. 1 udang vaname (*Litopenaeus vannamei*)

Sesuai dengan Peraturan Menteri Kelautan dan Perikanan Republik Indonesia Nomor 41 Tahun 2001, pada tanggal 12 Juli 2001, pemerintah secara resmi menjadikan udang vaname sebagai varietas unggulan oleh pembudidaya dalam negeri [26]. Dikenal dengan varietas unggulnya, udang vanamei dinilai memiliki beberapa kelebihan antara lain:

a. Ketahanan yang lebih baik terhadap penyakit;
b. Dapat tumbuh lebih cepat.
c. Ketahanan terhadap perubahan kondisi lingkungan;
d. Masa pemeliharaan relatif singkat, sekitar 90-100 hari per siklus;
e. Tingkat kelangsungan hidup (SR) atau standar hidup yang relatif tinggi.
f. Lebih hemat terhadap pakannya.

Berdasarkan laporan dari Kementerian Kelautan dan Perikanan pada tahun 2017-2018 udang menjadi komoditas utama yang berkontribusi besar pada volume ekspor hasil perikanan Indonesia. Komoditas tersebut mengalami peningkatan dengan kenaikan rata-rata volume 9.50% ton [3]. Sedangkan nilai ekspor udang mengalami penurunan dengan kenaikan rata-rata nilai ekspor sebesar -0.21%, kendati demikian nilai ekspor udang menunjukkan nilai terbesar dibandingkan dengan komoditas utama ekspor lainnya. Penurunan tersebut antara lain dipengaruhi oleh berbagai macam faktor seperti faktor cuaca, nilai harga jual dipasar global, kendala budidaya yang disebabkan oleh hama dan patogen (penyebab penyakit) serta faktor lainnya [4] [5] [1] [2] [6].

2.2.2 *Infectious Myonecrosis Virus* (IMNV)

Infectious myonecrosis virus (IMNV) merupakan penyakit utama pada budidaya udang [27]. Menurut Koesharyan pada tahun 2019 diketahui, penyakit tersebut dapat menginfeksi sebagian besar udang vaname yang mengakibatkan kematian besar dan dapat menyebabkan kematian 100% jika penanganannya terlambat dan panen awal serta akibatnya kerugian produksi. Semua ukuran udang dapat terinfeksi oleh IMNV baik larva maupun udang dewasa [7]. Penyakit IMNV dapat dilihat pada gambar 2.2.

Gambar 2. 2 Udang vaname yang terserang gejala IMNV

Udang yang terserang IMNV dapat merusak jaringan dan mengubah warna tubuhnya menjadi putih seperti kapas yang dapat menyebabkan terjadinya

perubahan warna tubuh udang menjadi seperti putih kapas. Pada awalnya, otot tubuh udang berwarna keputihan, biasanya akan terlihat pada siang hari dan kemudian keesokan harinya ruas terakhir pada tubuh udang berubah menjadi kemerahan. Pada umumnya penyakit ini disebabkan oleh tingkat kepadatan udang terlalu tinggi, dan pengaruh kualitas lingkungan yang memburuk, terutama stratifikasi suhu dan salinitas [9] [1]. Udang yang terinfeksi IMNV perlahan kehilangan nafsu makannya, menyebabkan kematian, dan tingkat kematian udang yang terinfeksi virus ini bisa mencapai 40 hingga 70% [10].

Hingga saat ini, metode pengobatan infeksi oleh virus tersebut belum ditemukan sehingga usaha yang dapat dilakukan yaitu dengan pencegahan. Salah satu metode pencegahan yaitu pengetahuan mengenai status penyebaran IMNV [11]. Menurut laporan yang terdata *food and agriculture organization* (FAO) pada tahun 2019 dapat diketahui penyebaran penyakit tersebut melalui serangkaian prosedur setelah diagnosis awal berdasarkan tanda-tanda klinis kasar. Ini termasuk demonstrasi histologis lesi, yang biasanya terlalu lambat untuk memungkinkan keputusan manajemen praktis oleh petani udang [8].

Selain itu, cara lain untuk dapat mencegahnya yaitu dengan pendekteksian virus menggunakan metode *polymerase chain reaction* (PCR) dan analisa histopatologi. Dikarenakan harganya yang mahal, pengujian menggunakan PCR hanya bisa dilakukan di tempat-tempat tertentu, seperti Balai Pengembangan Budidaya Air Payau (BBPBAP) [12].

2.2.3 Pengolahan Citra Digital

Pengolahan citra digital mengacu pada pemrosesan komputer digital. Citra digital terdiri dari sekumpulan elemen dengan lokasi dan nilai tertentu. Elemen-elemennya disebut dengan *picture elements, image elements*, dan *pixels*. *Pixel* adalah istilah yang paling umum digunakan untuk elemen gambar digital.

Pemrosesan gambar digital mencakup berbagai aplikasi yang diajukan dan luas. kadang-kadang perbedaan dibuat dengan mendefinisikan pemrosesan gambar sebagai suatu disiplin di mana input dan ouput suatu proses adalah gambar.

Tidak ada batas yang jelas dalam kontinum dari pemrosesan gambar di satu ujung ke visi komputer di ujung lainnya. Namun, salah satu paradigma yang berguna

adalah mempertimbangkan tiga jenis proses terkomputerisasi adalah sebagai berikut [28]:

a. *Low-level*

   Proses *low-level* meliputi operasi dasar seperti image *preprocessing* diantaranya yaitu *reduce noise* (mengurangi gangguan), c*ontrast enhancement* (peningkatan kontras), dan *image sharpening* (penajaman gambar). Hasil yang didapatkan dari *low-level* ini berupa gambar.

b. *Mid-level*

   Pemrosesan tingkat menengah melibatkan segmentasi (membagi gambar menjadi area atau objek). Objek dideskripsikan sebagai direduksi menjadi tampilan dan klasifikasi objek yang diinginkan. Masukan (*input*) ke proses antara adalah gambar, dan keluaran (*output*) dapat menghasilkan atribut yang berasal dari gambar (misalnya, tepi, kontur, identitas objek tertentu).

c. *High-level*

   Bentuk perawatan ini memberikan makna dari berbagai objek yang dikenali untuk menampilkan fungsi kognitif yang umumnya terkait dengan penglihatan. [29].

Berdasarkan penjelasan diatas, bahwa tempat logis dari tumpang tindih antara pemrosesan gambar dan analisis gambar adalah area pengenalan masing-masing daerah atau objek dalam suatu gambar. Dengan demikian, apa yang disebut dengan pemrosesan gambar digital mencakup proses *input* dan *output*-nya adalah gambar.

Selain itu, pemrosesan gambar digital mencakup proses mengekstraksi atribut dari gambar, hingga pengakuan objek individu. Proses memperoleh gambar dari area yang berisi teks, memproses ulang gambar itu, mengekstraksi (mensegmentasi) karakter individu, menggambarkan karakter dalam dari yang cocok untuk pemrosesan komputer, dan mengenali karakter individu tersebut dalam ruang lingkup apa yang disebut pengolahan citra digital.

2.2.4 Ruang Warna

Ketika manusia melihat sebuah gambar yang didapati, warna merupakan fitur yang paling terlihat untuk dapat diterima atau dirasakan. Adapun beberapa ruang warna

yang dapat merepresentasikan sebuah gambar yaitu RGB dan HSV. Ruang warna tersebut akan dijelaskan dibawah ini [30].

2.2.4.1 Ruang Warna RGB

RGB merupakan salah satu model ruang warna yang memiliki konsep dasar penambahan intensitas cahaya dasar (primer) yaitu merah (*red*), hijau (*green*), dan biru (*blue*). Ruang warna ini berdasarkan hasil akuisisi frekuensi warna oleh sensor elektronik yang dijadikan sebagai warna standar. Outputnya berupa sinyal analog yang memiliki intensitas disetiap warna yang dikodekan dalam 8 bit seperti yang tertera pada gambar 2.3.

Gambar 2. 3 Ruang warna RGB

Dari gambar 2.3 diatas merepresentasikan citra 24 bit masing-masing memiliki 8 bit untuk R, 8 bit untuk G, dan 8 bit untuk B.

2.2.5 Praproses (*Preprocessing*)

Praproses data merupakan langkah penting dalam proses menemukan pengetahuan. Hal tersebut dikarenakan keputusan kualitas harus didasarkan pada data yang berkualitas. Teknik pra-pemrosesan data meningkatkan kualitas data, meningkatkan akurasi dan efisiensi proses ekstraksi selanjutnya. Mendeteksi anomali data, memperbaikinya lebih awal, dan mengurangi data yang akan dianalisis dapat menghasilkan hadiah besar untuk pengambilan keputusan [31].

Preprocessing data mencakup kegiatan seperti mengurangi kebisingan, memilih himpunan bagian yang sesuai dari data yang dikumpulkan, dan menentukan subset fitur yang tepat yang menggambarkan konsep yang akan dipelajari. Data bersih ini akan menjadi input untuk algoritma dan implementasi tertentu. Oleh karena itu, data dan latar belakang pengetahuan, jika ada, mungkin perlu dijelaskan dan diformat dengan cara yang sesuai dengan persyaratan algoritma yang digunakan [32]. Pentingnya *preprocessing* data:

a. data *real* bisa menjadi kotor dan dapat mendorong ke ekstraksi pola / aturan yang tidak berguna.
b. pemrosesan data dapat menghasilkan kumpulan data yang lebih kecil dari aslinya, yang memungkinkan kami untuk meningkatkan efisiensi dalam proses penambangan data.
c. tidak ada data berkualitas, tidak ada hasil penambangan berkualitas.

2.2.5.1 Proses *Image Enhancement*

Peningkatan citra (*enhancement*) adalah proses pengolahan citra untuk menghasilkan hasil yang lebih baik dari citra asli pada pengaplikasian tertentu.. Kata spesifik penting di sini, karena itu menetapkan pada awalnya bahwa teknik peningkatan berorientasi masalah [33]. Dalam kasus ini, proses *enhancement* merupakan proses meningkatkan kualitas citra dengan mempertajam kontras yang dimiliki citra.

Sebuah citra yang tidak jelas ketajamannya akan menjadi kurang jelas, sehingga diperlukan adanya penajaman citra yang dapat membuat kualitas pada citra menjadi lebih baik. Adapun salah satu cara untuk menghasilkan citra yang memiliki kualitas yang tinggi yaitu dengan *high pass filter* [34]. *High-pass* filter adalah filter yang mentransmisikan frekuensi tinggi dari suatu gambar dan meningkatkan frekuensi rendah. Gambar yang difilter memiliki lebih sedikit pergeseran skala abu-abu di area "*smooth*" dan memberi tekanan pada tingkat abu-abu transisi (tepi) [29]. Proses *image enhancement* dapat dilihat pada gambar 2.4.

Gambar 2. 4 Proses *enhancement* dengan *high pass filter*

2.2.5.2 Proses Segmentasi dengan Operasi Morfologi

Segmentasi citra adalah proses membagi citra digital menjadi beberapa area (area), yang terdiri dari kumpulan piksel. Segmentasi citra digunakan untuk menemukan

objek yang memiliki batas bentuk objek, seperti garis dan kurva. Segmentasi bertujuan untuk menyederhanakan citra menjadi sesuatu yang lebih bermakna agar lebih mudah dianalisis.

Penggunaan operasi morfometrik umumnya didasarkan pada bentuk segmen atau wilayah citra sebagai objek yang diterapkan pada citra biner. Operasi dasar yang dipakai dalam pemrosesan operasi morfologi diantaranya meliputi: dilasi, erosi, closing dan opening [35].

Teknik *opening* biasanya menghaluskan kontur objek dengan menghilangkan area kecil. Teknik (*closing*) lebih cenderung menghaluskan kontur bentuk, tetapi (*closing*) adalah kebalikan dari (*opening*). Biasanya teknik (*closing*) menghilangkan serpihan sempit dan buah yang panjang dan tipis, menghilangkan lubang kecil, dan mengisi celah pada kontur gambar [36].

Citra yang telah disegmentasi akan diberi *masking* yang memiliki fungsi agar latar belakang putih yang terdapat pada citra tidak terhitung sebagai objek pada proses ekstraksi warna dan tekstur. Hasil dari proses segmentasi dengan operasi morfologi dapat dilihat pada gambar 2.5.

Gambar 2. 5 Proses segmentasi citra dengan *thresholding*

2.2.5.3 Proses Konversi Citra RGB ke *Grayscale*

Proses selanjutnya dalam *preprocessing* gambar adalah konversi gambar warna menjadi gambar tingkat abu-abu untuk membuat proses pemrograman lebih mudah karena jumlah bit dalam gambar skala abu-abu (8-bit) kurang dari jumlah bit dalam gambar RGB (24-bit). Persamaan 2.1 digunakan untuk mengubah citra RGB menjadi citra *grayscale* [37]. Hasil dari pemrosesan konversi citra RGB ke *grayscale* dapat dilihat pada gambar 2.6.

$$grayscale = 0.299 * R + 0.587 * G + 0.114 * B \quad (2.1)$$

dimana:

gray-scale: nilai tingkat abu-abu setelah konversi

R : nilai matriks komponen *red* (merah) pada ruang warna RGB

G : nilai matriks komponen *green* (hijau) pada ruang warna RGB

B : nilai matriks komponen *blue* (biru) pada ruang warna RGB

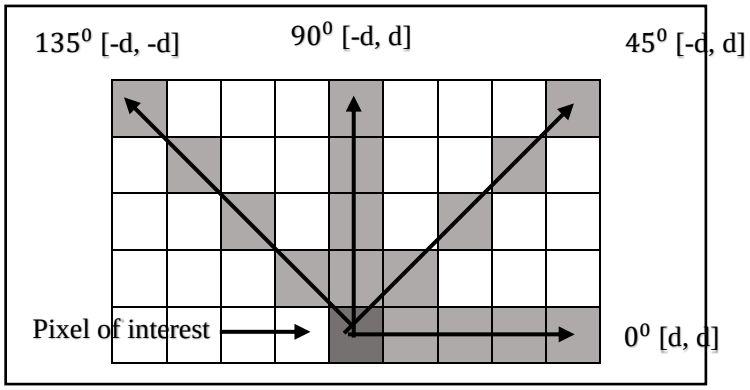
Gambar 2.6 Proses konversi citra RGB ke *grayscale*

2.2.6 *Gray Level Co-Occurrence Matrix* (GLCM)

Gray level co-occurrence matrix (GLCM) adalah metode yang digunakan untuk analisis tekstur / ekstraksi ciri. GLCM adalah matriks yang mewakili frekuensi di mana pasangan dua piksel dengan intensitas tertentu terjadi pada jarak dan arah tertentu dalam sebuah gambar [36].

Indeks GLCM awalnya dikembangkan oleh ilmuwan komputer yaitu Robert Haralick untuk kuantifikasi pola permukaan yang kasar dan kekasaran yang ditunjukkan dalam citra digital. Setiap indeks dapat menyoroti properti tekstur tertentu, seperti kehalusan, kekenyalan, dan ketidakteraturan [38]. Tekstur dapat dikatakan sebagai istilah yang digunakan untuk mengkarakterisasikan variasi level nada abu-abu dalam sebuah gambar [39].

Kookurensi dapat dipahami sebagai kejadian bersama, yaitu jumlah kemunculan pada piksel yang berdekatan dengan nilai piksel lain sebagai fungsi jarak (d) dan arah sudut (Ɵ). Jarak dinyatakan dalam piksel dan arah dinyatakan dalam derajat. Orientasi tersebut terdiri dari empat sudut. Dimana sudut tersebut dengan interval 45^0, yaitu 0^0, 45^0, 90^0 dan 135^0 dengan jarak antar piksel yang ditentukan sebesar 1 piksel [40]. Keempat arah tersebut seperti pada gambar 2.7.



Gambar 2.7 Orientasi sudut dan jarak

Pada gambar 2.8 GLCM dapat mudah untuk dikonstruksi. Pertama kali skala abu-abu gambar dalam bentuk matriks didiskritkan menjadi matriks integer dengan membagi rentang nilai piksel kontinu ke dalam N *bin* dengan lebar yang sama, yang disebut dengan level (tingkat) abu-abu. Kemudian nilai dalam sebuah *bin* tersebut dipetakan ke dalam satu level abu-abu seperti pada gambar 2.8 a-b.

Elemen-elemen GLCM dihitung berdasarkan gambar diskrit tersebut dengan menghitung seberapa sering pasangan piksel dengan tingkat abu-abu tertentu dan dalam hubungan spasial tertentu terjadi dalam matriks seperti pada gambar 2.8 c. Kedekatan piksel tersebut dapat didefinisikan dalam 4 sudut berbeda yaitu: 0^0, 45^0, 90^0 dan 135^0 seperti pada gambar 2.8 d. Kedekatan piksel tersebut dapat diilustrasikan pada derajat nol (piksel yang berdekatan dengan sisi kanan) yang dipilih seperti pada gambar 2.8.

GLCM kemudian dinormalisasi untuk dapat membuat jumlah semua elemen sama dengan satu seperti pada gambar 2.8 e. Elemen generik dari matriks tersebut dicatat sebagai g(i, j). Berdasarkan jumlah level abu-abu dan elemen matriks yang dinormalisasi, indeks GLCM dapat dihitung seperti pada gambar 2.8 f. Untuk tujuan analitis, indeks GLCM dihitung dengan aturan kedekatan yang berbeda (misalnya pada sudut yang berbeda) dan lebar bin sering digunakan bersamaan [41].

Original Percentage (%) Image (5x5 pixel image)

63	45	70	90	99
11	27	24	30	85
7	13	10	21	53
5	10	17	37	61
0	3	12	33	48

(a)

Discretized Image (Bin Width = 20)

(60,79)	(40,59)	(60,79)	(80,99)	(80,99)
(0,19)	(20,39)	(20,39)	(20,39)	(80,99)
(0,19)	(0,19)	(0,19)	(20,39)	(40,59)
(0,19)	(0,19)	(0,19)	(20,39)	(60,79)
(0,19)	(0,19)	(0,19)	(20,39)	(40,59)

(b)

+1 +1 +1

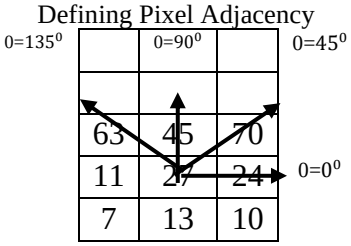
GLCM (5 intervals, N_g=5, 0=0^0)

g(i,j)	j=0 (0,19)	j=1 (20,39)	j=2 (40,59)	j=3 (60,79)	j=4 (80,99)
i=0 (0,19)	6	4	0	0	0
i=1 (20,39)	0	2	2	1	1
i=2 (40,59)	0	0	0	1	0
i=3 (60,79)	0	0	1	0	1
i=4 (80,99)	0	0	0	0	1

Gray level

(c)

Different Angles Defining Pixel Adjacency



(d)

Normalized GLCM

g(i,j)	j=0 (0,19)	j=1 (20,39)	j=2 (40,59)	j=3 (60,79)	j=4 (80,99)
i=0 (0,19)	0.3	0.2	0	0	0
i=1 (20,39)	0	0.1	0.1	0.05	0.05
i=2 (40,59)	0	0	0	0.05	0
i=3 (60,79)	0	0	0.05	0	0.05
i=4 (80,99)	0	0	0	0	0.05

Gray level

(e)

contoh : GLCM Contrast

$$contrast = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (i-j)^2 . g^2(i-j)$$

$$\begin{aligned} contrast = {} & (0-0)^2.0.3^2 + (0-1)^2.0.2^2 + (0-2)^2.0^2 + (0-3)^2.0^2 + (0-4)^2.0^2 \\ & +(1-0)^2.0^2 + (1-1)^2.0.1^2 + (1-2)^2.0.1^2 + (1-3)^2.0.05^2 + \\ & (1-4)^2.0.05^2 \\ & +(2-0)^2.0^2 + (2-1)^2.0^2 + (2-2)^2.0^2 + (2-3)^2.0.05^2 + (2-4)^2.0^2 \\ & +(3-0)^2.0^2.(3-1)^2.0^2 + (3-2)^2.0.05^2 + (3-3)^2.0^2 + \\ & (3-4)^2.0.05^2 \\ & +(4-0)^2.0^2 + (4-1)^2.0^2 + (4-2)^2.0^2 + (4-3)^2.0^2 + (4-4)^2.0^2 \\ = {} & 0.09 \end{aligned}$$

(f)

Gambar 2.8 Ilustrasi perhitungan indeks GLCM untuk gambar sampel [47]

Langkah-langkah yang dilakukan untuk menghitung GLCM adalah sebagai berikut:

a. Inisiasi dilakukan dengan membentuk matriks GLCM dari sepasang dua piksel yang disejajarkan pada arah 0°, 45°, 90° atau 135° secara berurutan.
b. enambahkan nilai transposnya dengan nilai matriks GLCM asli (awal) untuk membentuk matriks simetris.
c. Menormalisasikan larik GLCM dengan membagi setiap elemen larik dengan jumlah pasangan pikselnya [42].
d. Ekstraksi ciri. GLCM digunakan sebagai indeks ekstraksi ciri yang digunakan dalam penelitian ini pada orde kedua sebanyak 4 fitur ekstraksi tekstur pada GLCM [43], yaitu :

1. *Angular Second Moment* (ASM)

   Ukuran keseragaman gambar atau nilai yang digunakan untuk menghitung kerapatan intensitas pasangan dalam matriks. ASM dapat dikatakan menunjukkan mewakili ukuran homogenitas (keseragaman) dari suatu citra, dihitung dengan persamaan 2.2 sebagai berikut:

$$ASM = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} g^2(i.j) \tag{2.2}$$

2. *Energy*

   Mengukur keseragaman tekstur, atau pengulangan pasangan piksel. Energi tinggi terjadi ketika distribusi nilai tingkat abu-abu konstan atau berkala, dihitung dengan persamaan 2.3 sebagai berikut:

$$energy = \sqrt{\sum_{i=0}^{N_g-1} \quad \sum_{j=0}^{N_g-1} g^2(i.j)} \tag{2.3}$$

3. *Correlation*

   Mengukur ketergantungan linear pada gambar. Nilai korelasi tinggi menyiratkan hubungan linier antara tingkat abu-abu pasangan piksel yang berdekatan, dihitung dengan persamaan 2.4 sebagai berikut:

$$correlation = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} \frac{(\mathrm{i}-\mu).(\mathrm{j}-\mu).g(i,j)}{\sigma^2} \tag{2.4}$$

   dimana :

   - μ (*Mean)* untuk mengukur rata-rata nilai level abu-abu dalam suatu gambar, dihitung dengan persamaan 2.5 sebagai berikut:

$$mean = \sum_{i=0}^{N_g-1} \quad \sum_{j=0}^{N_g-1} \text{i}.\, g\,(i,j) \tag{2.5}$$

- *Variance* merupakan ukuran heterogenitas, varians meningkat ketika nilai tingkat abu-abu berbeda dari rata-rata mereka, dihitung dengan persamaan 2.6 sebagai berikut:

$$variance = \sum_{i=0}^{N_g-1} \quad \sum_{j=0}^{N_g-1} (i-\mu)^2 . g(i,j) \tag{2.6}$$

4. *Homogenity*

   Mengukur homogenitas gambar. Peka terhadap keberadaan elemen diagonal dekat dalam GLCM, mewakili kesamaan tingkat abu-abu antara piksel yang berdekatan, dihitung dengan persamaan 2.7 sebagai berikut:

$$homogenity = \sum_{i=0}^{N_g-1} \quad \sum_{j=0}^{N_g-1} \frac{1}{1+(i-j)^2} . g(i,j) \tag{2.7}$$

   dimana :

   N_g : jumlah tingkat abu-abu (*gray levels*)

   i : baris

   j : kolom

   g : pasangan matriks *intensitas co-occurrence* [38]

2.2.7 *Color Moment*

Color moment merupakan suatu metode pengukuran yang dapat digunakan untuk membedakan gambar berdasarkan fitur warna yang terdapat didalamnya. Setelah dihitung, momen-momen warna tersebut memberikan pengukuran untuk kesamaan warna antar gambar. dasar momen warna diasumsikan bahwa distribusi warna dalam gambar dapat diartikan sebagai distribusi probabilitas [44].

Color moment merupakan pilihan yang dapat digunakan untuk mengekstraksi fitur warna pada udang vaname yang sehat dengan udang vaname yang terserang gejala penyakit IMNV. Ruang warna yang akan digunakan yaitu HSV dengan komponen warna yang dimiliki terdiri dari (*hue, saturation, value*). Gambar akan dibagi menjadi 9 *moment* berupa 3 *moments* untuk setiap 3 *color channels* pada HSV sehingganya menghasilkan 9 ciri. Kemudian 9 ciri ini akan menjadi ciri warna dari gambar yang digunakan tersebut. Secara umum, *color moment* menggunakan 3 fitur, yaitu

a. *Mean* (μ_c)

Mean mewakili nilai rata-rata untuk setiap saluran warna. Mean adalah nilai rata-rata piksel (P_{ij}) untuk masing-masing saluran R, G dan B seperti pada persamaan 2.8 sebagai berikut.

$$\mu_c = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} p_i^c \tag{2.8}$$

b. Standar Deviasi (σ_c)

Standar deviasi mewakili akar dari *variance* [45] yang dapat dihitung dengan menggunakan persamaan 2.9 sebagai berikut.

$$\sigma_c = \left[\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p_{ij}^c - \mu_c\right)^2\right]^{\frac{1}{2}} \tag{2.9}$$

c. *Skewness* (θ_c)

Skewness mewakili pengukuran yang digunakan untuk simetri [45] yang dapat dihitung dengan menggunakan persamaan 2.10 sebagai berikut.

$$\theta_c = \left[\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p_{ij}^c - \mu_c\right)^3\right]^{\frac{1}{3}} \tag{2.10}$$

dimana :

N : dimensi lebar citra

M : dimensi tinggi citra

(P_{ij}) : nilai intensitas warna pada baris *i* dan kolom *j*

2.2.8 *K-Fold Cross-Validation*

Cross-validation adalah metode statistik untuk mengevaluasi dan memverifikasi keakuratan algoritma pelatihan untuk model yang dibangun di atas kumpulan data tertentu. [46]. Model dibangun untuk tujuan memprediksi atau mengklasifikasikan data baru yang tidak ada dalam dataset. Data yang digunakan tersebut dibagi menjadi dua segmen, yaitu: satu digunakan untuk melatih suatu model yang dibuat selama proses disebut data latih dan yang lainnya digunakan sebagai validasi model yang disebut dengan data tes.

K-fold cross-validation merupakan salah satu metode yang terdapat dalam validasi silang yang digunakan untuk menghitung akurasi prediksi dari suatu sistem yang sedang dibangun. Dengan *k-fold cross-validation*, data pada awalnya dibagi

menjadi k bagian atau potongan dengan rasio ukuran yang sama (atau hampir sama). Selain itu, pelatihan dan validasi dilakukan sebanyak k kali, setiap iterasi menggunakan segmen yang berbeda sebagai data uji atau validasi, k-1 yang tersisa diambil sebagai data pelatihan dan hasil dari setiap iterasi dirata-ratakan [47]. Data biasanya bertingkat sebelum dipisah menjadi beberapa bagian *k-fold*.

Stratifikasi merupakan proses melakukan penataan ulang data untuk dapat memastikan setiap kelipatan merupakan perwakilan yang baik dari jumlah keseluruhan data yang ada. Metode *k-fold cross-validation* melakukan generalisasi data tersebut dengan mensegmentasikannya ke dalam sebuah k partisi yang berukuran sama. Selama proses tersebut, salah satu dari partisi dipilih untuk *training*, sedangkan sisanya untuk percobaan. Prosedur tersebut diulangi sebanyak k kali sehingganya setiap partisi dapat digunakan untuk testing tepat satu kali.

Sebagai contoh dalam masalah klasifikasi biner di mana setiap kelas terdiri dari 50% dari data, yang terbaik adalah mengatur data sedemikian rupa sehingga di setiap flip, setiap kelas terdiri dari sekitar setengah *instance*. Untuk memberikan gambaran mengenai *k-fold cross-validation* dapat dilihat pada tabel 2.2. Berikut merupakan contoh dari *k-fold cross-validation* dengan k=5.

"dataset=A1,A2,A3,A4,A5"

Tabel 2.2 contoh *k-fold cross-validation*

percobaan ke-	Data latih	Data tes
1	A2,A3,A4,A5	A1
2	A1,A3,A4,A5	A2
3	A1,A2, A4,A5	A3
4	A1,A2,A3, A5	A4
5	A1,A2,A3,A4	A5

Semisal dalam sebuah penelitian yang dilakukan yaitu 5 *k-fold cross-validation*, maka dilakukan iterasi sebanyak 5 kali pada dataset dengan mengambil data untuk segmen proses pengujian (*testing*) pada setiap 1 segmen dan segmen sisanya dijadikan sebagai segmen proses pelatihan (*training*) dengan k-1 segmen pada setiap iterasinya dilakukan iterasi sebanyak 5 kali. Dengan data yang digunakan

berjumah 1923 data, maka data latihnya akan berjumlah 1538 data latih dan data tes berjumlah 385 data.

2.2.9 *Support Vector Machine* (SVM)

Beberapa permasalahan yang ditimbulkan dari pemodelan data empiris yaitu data yang diperoleh bersifat heterogen dan dapat mengarah pada analisis dengan pendekatan (*traditional neural network*) dengan sifat dimensi tinggi (ruang fitur) Sulit untuk digeneralisasi yang menghasilkan model (*overfit*) data [48].

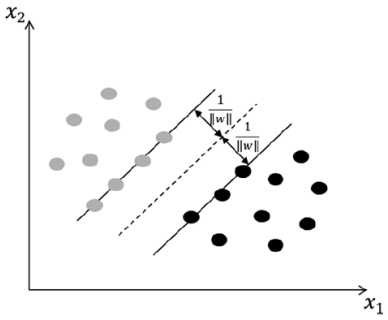
Support vector machine (SVM) adalah sistem pembelajaran yang menggunakan ruang sebagai fungsi linier dalam ruang fitur multidimensi yang dibentuk dengan menggunakan algoritma pembelajaran berdasarkan teori optimasi dengan (*learning*) bias [49] dan juga metode yang relatif baru untuk membuat prediksi untuk klasifikasi dan regresi.

SVM dikembangkan untuk memecahkan masalah klasifikasi karena memiliki kemampuan generalisasi data yang lebih baik daripada metode yang tersedia sebelumnya. Untuk menemukan kemampuan generalisasi yang baik yaitu memilih beberapa subset (kecil) dari data pelatihan, yang disebut dengan *support vector* (SV). Kemudian metode SV ini memungkinkan untuk memperluas fungsi dimensi tinggi menggunakan basis kecil yang dibangun dari SV. Representasi fungsi jenis baru ini membuka peluang baru untuk menyelesaikan berbagai masalah perkiraan dan estimasi fungsi [50].

Salah satu manfaat yang didapatkan dengan menggunakan pendekatan metode SVM yaitu Model yang dibangun tidak hanya dapat membantu menjelaskan model *support vector* (SV), tetapi juga dapat secara eksplisit bergantung pada subset dari (titik data). Prinsip dasar penggunaan SVM adalah mencari lapisan terbaik (*hyperplane*) yang bertindak sebagai pemisah dua kelas dalam ruang *input* (masukan) melalui strategi yang disebut dengan *structural risk minimization* [51]. *Hyperplane* dapat berasimilasi dengan garis yang terdapat di dua dimensi dan *flat plane* yang terdapat di *multiple plane*. SVM melatih menggunakan kumpulan data pelatihan dan merupakan metode baru. Ini adalah metode agregasi dan prediksi dari data pembelajaran mesin.

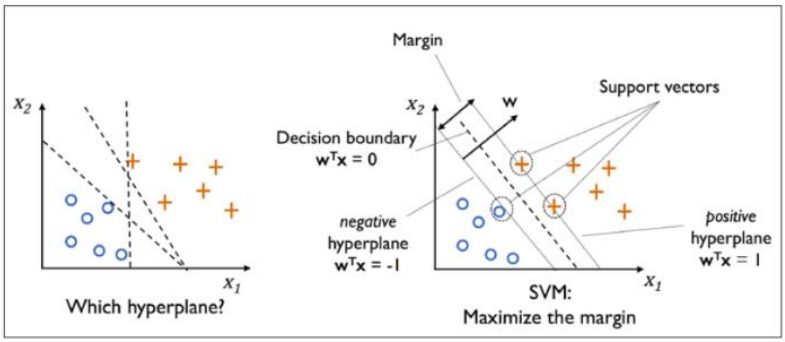
2.2.9.1 *Hard-Margin* SVM (*Linear* SVM)

Metode SVM merupakan sebuah teknik klasifikasi yang digunakan untuk menemukan *hyperplane* dengan dua kelas yang terpisah secara *linear* dengan mengklasifikasikan data seperti pada gambar 2.9.



Gambar 2.9 *Hard Margin Hyperplane* SVM [58]

Dari gambar 2.9 diatas, dapat diketahui bahwa fungsi $g(x) = w^T x + b$ dapat memisahkan dua kelas dengan *margin* maksimum yang dimiliki dapat memberikan sebuah gambaran untuk kelas positif dan kelas negatif yang terpisah secara total dengan lingkaran berwarna abu – abu yang berada dekat dengan garis x_2 sedangkan untuk lingkaran berwarna hitam terletak lebih dekat dengan garis x_1. Gambar 2.9 juga menggambarkan *hyperplane* pemisah tersebut sesuai dengan SVM *hard-margin* (juga disebut SVM linier) [52].



Gambar 2.10 *Hyperplane* terbaik yang memisahkan antara dua kelas positif (+1) dan negatif (-1) [59]

Pada gambar 2.10 diatas dapat diperlihatkan beberapa *pattern* yang terdiri dari dua buah kelas yaitu positif (+1) dan negatif (-1).

Untuk dapat menemukan *hyperplane* pemisah terbaik yaitu dengan mengukur mengukur margin *hyperplane* dan mencari titik maksimal yang dimiliki. Jarak antara *hyperplane* dengan data yang terdekat dari masing-masing setiap kelas

disebut dengan *margin*, sedangkan dari subset dataset pelatihan yang paling dekat disebut dengan *support vector* [53].

Dari gambar 2.9 dapat diketahui bahwa *hyperplane* yang terbaik terdapat tepat ditengah kedua kelas antara *hyperplane* positif dan *hyperplane* negatif yang ditandai dengan garis putus-putus. Sedangkan yang dinamakan *support vector* diberi tanda positif berwarna hijau dan bulat berwarna merah yang berada dalam lingkaran hitam.

Inti proses dari metode SVM yaitu upaya mencari lokasi *hyperplane* yang optimal. Dapat diasumsikan sebuah data *learning* (pembelajaran) dengan *datapoints* dinotasikan x_i $(i = 1,.,m)$ memiliki label dua kelas $y_i = \pm 1$ yaitu kelas positif $(+1)$ dan kelas negatif (-1) sehingga *decision function* akan didapatkan pada persamaan 2.11 berikut.

$$f(x) = sign(w.x + b) \tag{2.11}$$

dimana :

w : vektor bobot

x : nilai masukan *datapoint*

b : bias

"$\cdot$" : skalar atau *inner product* sehingga $w.x = w^T x$

Dari hasil *decision function* diatas, data dapat diklasifikasikan dengan tepat jika $y_i(w.x_i + b) > 0$ karena $w.x_i + b$ dimana ketika $(y_i = +1)$ maka nilai harus bernilai positif, dan $(y_i = -1)$ maka nilai harus negatif. *Decision function* akan menjadi tidak berubah (*invarian*) ketika proses penskalaan positif baru dilakukan argumen ke persamaan fungsi. Hal ini dapat menimbulkan ambiguitas ketika mendefinisikan konsep jarak (*margin*).

Oleh karena itu perlu ditentukannya penskalaan untuk w, b dengan menetapkan $w.x + b = 1$ untuk titik terdekat di satu sisi dan $w.x + b = -1$ untuk terdekat di sisi lainnya. *Hyperplane* yang melewati $w.x + b = 1$ dan $w.x + b = -1$ disebut sebagai *hyperplane* kanonik, sedangkan wilayah diantara *hyperplane* kanonik disebut sebagai *margin band* [54].

Untuk dapat memperoleh nilai margin terbesar yaitu memaksimalkan nilai jarak antara *hyperplane* dengan titik terdekat, yaitu $\frac{1}{||w||}$. Hal ini disebut sebagai *Quadratic Programming* (QP) *problem*, yaitu dilakukan pencarian titik minimalnya pada persamaan 2.12 berikut.

$$\min \tau\,(w) = \frac{1}{2}\left|\left|w\right|\right|^2 \tag{2.12}$$

dengan memperhatikan subjek *constraint* (kendala) pada persamaan 2.13 sebagai berikut.

$$y_i(w.x_i + b) \geq 1, \quad i = 1,2,3 \ldots, n \tag{2.13}$$

dimana :

x_i : nilai masukan *datapoint*

y_i : *output* dari *datapoint* x_i (target kelas yang memiliki label)

Persamaan di atas adalah masalah optimasi yang meminimalkan fungsi objek dalam persamaan (2.12) dengan batasan kendala dalam persamaan (2.13). Permasalahan tersebut dapat direduksi dengan berbagai teknik komputasi yaitu menggunakan fungsi *lagrange*. Fungsi pengali adalah jumlah dari fungsi tujuan objektif dan kendala m dikalikan dengan fungsi (*lagrange multiplier*). pada persamaan 2.14 sebagai berikut.

$$L(w,b) = \frac{1}{2}(w.w) - \sum_{i=1}^{m} a_i(y_i(w.x_i + b) - 1) \tag{2.14}$$

dimana :

$L(w,b)$: fungsi *lagrange*

a_i : *lagrange multipliers* yang bernilai nol atau positif $(a_i \geq 0)$

Ketika nilainya minimum, itu dilakukan penurunan dari b dan w mengatur nilai menjadi 0 seperti persamaan 2.15 dan persamaan 2.16 berikut [54].

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} a_i\, y_i = 0 \tag{2.15}$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} a_i\, y_i x_i = 0 \tag{2.16}$$

dimana :

$\frac{\partial L}{\partial b}$: turunan parsial dari L (fungsi *lagrange*) terhadap b (bias)

$\frac{\partial L}{\partial w}$: turunan parsial dari L (fungsi *lagrange*) terhadap w (bobot vektor)

Substitusi nilai w dari persamaan (2.17) kedalam bentuk $L(w, b)$ sehingga akan diperoleh formulasi ganda yang juga dikenal dengan *wolfe dual.*

$$W(a) = \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i,j=1}^{m} a_i\, a_j y_i y_j (x_i, x_j) \tag{2.17}$$

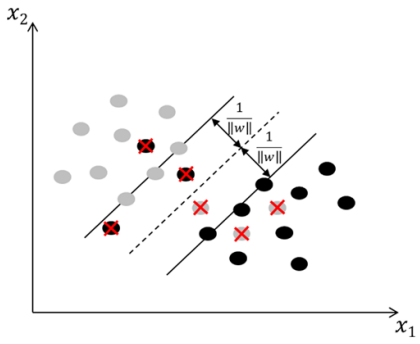
Dimana nilai a_i terhadap kendala sebagai berikut.

$$a_i \geq 0 \sum_{i=1}^{m} a_i\, y_i = 0 \tag{2.18}$$

Dari hasil perhitungan yang didapatkan kebanyakan memperoleh nilai positif. Data yang memiliki korelasi positif disebut (*support vector*).

2.2.9.2 *Soft-Margin* SVM

Jika data tidak dapat dipisahkan sepenuhnya, variabel *slack* x_i dimasukkan ke dalam fungsi SVM objektif untuk mentolerir kesalahan pada saat klasifikasi. Seperti halnya titik-titik yang ditandai oleh x pada gambar 2.11. Dalam hal ini, SVM tidak lagi menggunakan *hard margin classifier* yang sepenuhnya mengklasifikasikan semua data. Sehingga pada permasalahan ini *soft margin classifier* memungkinkan model untuk mengklasifikasikan titik-titik tertentu di sekitar batas pemisahan untuk mengklasifikasikan sebagian besar data dengan benar [52].



Gambar 2. 11 Beberapa kesalahan klasifikasi pada *soft-margin* SVM [58]

Berdasarkan pada gambar 2.11 diatas, dijelaskan bahwa data yang terdapat pada kedua kelas tersebut tidak sepenuhnya terpisah. Hal ini dapat dilihat pada beberapa lingkaran abu-abu yang memiliki sebaran di sekitar area lingkaran hitam dan sebaliknya ada lingkaran hitam yang memiliki sebaran di sekitar lingkaran abu-abu.

Perbedaan antara persamaan *soft margin* dan *hard margin* adalah terdapat perubahan kecil yang ditandai dengan adanya variabel *slack* $\varepsilon_i \geq 0$. Variabel *slack* merupakan sebuah variabel ukuran kesalahan pada klasifikasi seperti pada persamaan (2.19) seperti berikut.

$$(y_i(w.x_i + b) \geq 1 - \varepsilon_i), i = 1,2 \ldots, n \quad (2.20)$$

dimana :

ε_i : *slack* variabel

Kemudian ketika akan dilakukan minimasi penjumlahan error $\sum_{i=1}^{m} \varepsilon_i$ yaitu pada persamaan 2.21 sebagai berikut.

$$\min \tau\,(w) = \frac{1}{2}\left|\left|w\right|\right|^2 + c\sum_{i=1}^{m} \varepsilon_i \quad (2.21)$$

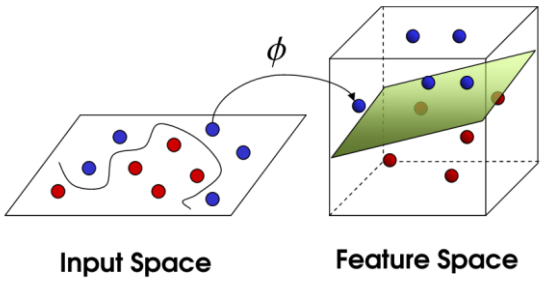
Parameter c digunakan sebagai pengontrol (*trade off*) antara margin dengan kesalahan klasifikasi [55].

2.2.9.3 Kernel SVM

Ketika masalah tidak dapat dipisahkan secara *linear* dalam ruang input, SVM *soft margin* tidak dapat menemukan *hyperplane* pemisah yang kuat untuk meminimalkan jumlah titik data yang tidak terklasifikasi dan yang menggeneralisasi dengan baik.

Jika masalahnya tidak dapat dipisahkan secara linier di ruang input, SVM *soft margin* tidak dapat menemukan pembatas yang kuat (*hyperplane*) untuk meminimalkan jumlah titik data yang tidak tergeneralisasi dan terklasifikasi dengan baik.

Untuk alasan ini, inti digunakan untuk mengubah data menjadi ruang dimensi yang lebih besar (tinggi), yang disebut ruang kernel, di mana data dapat dipisahkan secara linier. Dalam ruang kernel hyperplane linier dengan demikian dapat diperoleh untuk memisahkan kelas-kelas berbeda yang terlibat dalam tugas klasifikasi alih-alih menyelesaikan tatanan tinggi yang memisahkan *hypersurface* di ruang input. Ini adalah metode yang menarik, karena *overhead* pergi ke ruang kernel tidak signifikan dibandingkan dengan mempelajari permukaan *nonlinear*.



Gambar 2. 12 Transformasi dari *input space* ke *feature space* [63]

Pada gambar 2.12 menunjukkan bahwa jika data *input* tidak dapat didekomposisi secara linier, maka akan ditransformasikan ke dalam ruang dimensi yang lebih besar (ruang fitur). Linier *hyperplane* membentuk garis yang dapat dipisahkan antar kelas, sedangkan *hyperplane* nonlinier membentuk bidang yang dapat dipisahkan antar kelas.[47].

Dibawah ini contoh ilustrasi proses pemisahan data dengan kernel. Diketahui data berupa ruang masukan (*input space*) dengan dua buah = $\{x_1, x_2\}$ dan = $\{y_1, y_2\}$. Diasumsikan data tersebut akan dibuat fungsi K (kernel) dengan menggunakan masukan x dan y seperti berikut.

$$K(x,y) = (x^T y)^2 \tag{2.22}$$

$$K(x,y) = (x_1 y_1 + x_2 y_2)^2 \tag{2.23}$$

$$K(x,y) = (x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2)^2 \tag{2.24}$$

$$K(x,y) = \left(x_1^2, \sqrt{2}x_1 x_2, x_2^2\right)^T \left(y_1^2, \sqrt{2}y_1 y_2, y_2^2\right) \tag{2.25}$$

$$K(x,y) = \Phi(x)^T \Phi(y) \tag{2.26}$$

dimana :

K : fungsi kernel

x, y : *input space*

Φ : fitur pemetaan

Nilai K di atas secara implisit mendefinisikan pemetaan ke ruang berdimensi lebih tinggi sebagai berikut.

$$\Phi(x) = \left\{x_1^2, \sqrt{2}x_1 x_2, x_2^2\right\} \tag{2.27}$$

Setiap kernel k tersebut memiliki fitur pemetaan Φ yang terkait. Φ mengambil input $x \in X$ (*input space*) dan memetakannya ke dalam F (*feature space*). Kernel

$K(x, y)$ mengambil dua ruang input dan menyamakannya dalam ruang F (*feature space*) sebagai berikut.

$$\Phi: X \rightarrow F \tag{2.28}$$

$$K: X \text{ x } X \rightarrow \mathbb{R}, \quad K(x, y) = \Phi(x)^T . \Phi(y) \tag{2.29}$$

dimana :

$\Phi: X \rightarrow F$: fungsi Φ(fitur pemetaan) memetakan himpunan X(*input space*) ke dalam himpunan F (*feature space*)

X : *input space*

F : *feature space*

Dari fungsi di atas, beberapa perhitungan data menggunakan (ruang fitur) dilakukan dengan persamaan berikut [54].

$$f\big(\Phi(x)\big) = sign(w . \Phi(y) + b) \tag{2.30}$$

$$f\big(\Phi(x)\big) = sign(\textstyle\sum_{i=1}^{m} a_i \, y_i K(x, y) + b) \tag{2.31}$$

dimana:

b : nilai bias

m : *support vector*

K(x,y) : fungsi kernel

Agar nilai k dapat digunakan sebagai suatu fungsi kernel diharuskan memenuhi *mercer condition* diantaranya yaitu [56]:

- Nilai F (*feature space*) harus berupa vektor dengan *dot product* (F biasa disebut dengan *hilbert space)*
- Jika k adalah fungsi pasti positif maka nilai K harus benar
- Jika K_1 dan K_2 adalah fungsi kernel, maka :

$$K(x, y) = K_1(x, y) + K_2(x, y) : \textit{direct sum} \tag{2.32}$$

$$K(x, y) = \alpha K_1(x, y) : \textit{scalar product} \tag{2.33}$$

$$K(x, y) = K_1(x, y) K_2(x, y) : \textit{direct product} \tag{2.34}$$

Berikut adalah beberapa fungsi kernel umum yang digunakan oleh SVM.

a. *Linear* Kernel

Kernel linier adalah yang paling sederhana dari semua fungsi perkalian lainnya. *Linear* kernel digunakan pada data yang sudah dianalisis memiliki batas kelas *linear* atau sudah terpisah secara *linear* [57]. Kernel ini dinyatakan dalam persamaan 2.35 sebagai berikut.

$$K(x, y) = x^T y \tag{2.35}$$

Pemetaan fungsi Φ merupakan identitas atau tidak ada pemetaan.

b. *Radial Basis Function* (RBF) Kernel

Kernel RBF adalah kernel yang paling banyak digunakan untuk memecahkan masalah klasifikasi kumpulan data yang tidak dapat dipisahkan secara linier. [47]. Kernel dapat memetakan pola non-linear ke ruang dimensi yang lebih tinggi dan menangani kasus di mana hubungan antara label kelas dan atribut non-linear [58]. Kernel ini dinyatakan dalam persamaan 2.36 sebagai berikut.

$$K(x, y) = \exp\left[-\gamma\left||x - y|\right|^2\right] \tag{2.36}$$

Dalam penelitian ini, data latih dihitung menggunakan salah satu metode dengan metode sekuensial. Hasil dari *training* tersebut merupakan hasil pembelajaran yang akan dijadikan model yang disimpan menjadi acuan rujukan sistem dalam menentukan udang yang terkena penyakit IMNV atau tidak dari dataset baru. Adapun proses dari *training data* tersebut dilakukan sebagai berikut.

1. Menginisiasi awal nilai untuk parameter C dan γ.
   dimana :
   C (*complexity*) : digunakan dalam proses *training* yang nantinya memberikan batas nilai alfa
   γ (*gamma*) : digunakan untuk melakukan *controlling* kecepatan *training*
2. Memasukkan nilai data hasil ekstraksi warna dan tekstur dengan membaginya menjadi 2 bagian masing-masing label +1 untuk udang yang terjangkit gejala penyakit IMNV, sedangkan label -1 untuk udang normal (sehat).
3. Menentukan nilai *dot product* kernel setiap data dengan menggunakan persamaan 2.35 dan persamaan 2.36. Sebelum dilakukan perhitungan, data di

transpose terlebih dahulu karena menggunakan perkalian matriks AxA^T. Setiap data ini dibandingkan dengan dirinya sendiri dan dengan data lainnya. Semisal memiliki data latih berjumlah 10 dataset maka data tersebut direpresentasikan sebagai berikut seperti pada tabel 2.3.

Tabel 2. 3 Perbandingan 10 dataset

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	K(A1,A1)	K(A1,A2)	K(A1,A3)	K(A1,A4)	K(A1,A5)	K(A1,A6)	K(A1,A7)	K(A1,A8)	K(A1,A9)	K(A1,A10)
A2	K(A2,A1)	K(A2,A2)	K(A2,A3)	K(A2,A4)	K(A2,A5)	K(A2,A6)	K(A2,A7)	K(A2,A8)	K(A2,A9)	K(A2,A10)
A3	K(A3,A1)	K(A3,A2)	K(A3,A3)	K(A3,A4)	K(A3,A5)	K(A3,A6)	K(A3,A7)	K(A3,A8)	K(A3,A9)	K(A3,A10)
A4	K(A4,A1)	K(A4,A2)	K(A4,A3)	K(A4,A4)	K(A4,A5)	K(A4,A6)	K(A4,A7)	K(A4,A8)	K(A4,A9)	K(A4,A10)
A5	K(A5,A1)	K(A5,A2)	K(A5,A3)	K(A5,A4)	K(A5,A5)	K(A5,A6)	K(A5,A7)	K(A5,A8)	K(A5,A9)	K(A5,A10)
A6	K(A6,A1)	K(A6,A2)	K(A6,A3)	K(A6,A4)	K(A6,A5)	K(A6,A6)	K(A6,A7)	K(A6,A8)	K(A6,A9)	K(A6,A10)
A7	K(A7,A1)	K(A7,A2)	K(A7,A3)	K(A7,A4)	K(A7,A5)	K(A7,A6)	K(A7,A7)	K(A7,A8)	K(A7,A9)	K(A7,A10)
A8	K(A8,A1)	K(A8,A2)	K(A8,A3)	K(A8,A4)	K(A8,A5)	K(A8,A6)	K(A8,A7)	K(A8,A8)	K(A8,A9)	K(A8,A10)
A9	K(A9,A1)	K(A9,A2)	K(A9,A3)	K(A9,A4)	K(A9,A5)	K(A9,A6)	K(A9,A7)	K(A9,A8)	K(A9,A9)	K(A9,A10)
A10	K(A10,A1)	K(A10,A2)	K(A10,A3)	K(A10,A4)	K(A10,A5)	K(A10,A6)	K(A10,A7)	K(A10,A8)	K(A10,A9)	K(A10,A10)

	data pengujian
	data pelatihan

Dari tabel 2.3 diatas untuk mendapatkan hasil perhitungan dari setiap data dimisalkan dengan contoh K(A1,A1) diperoleh menggunakan persamaan 2.37 sebagai berikut.

$$K(A1,A1) = \big((A1 * A1) + (A1 * A2) + (A1 * A3) + (A1 * A4) + (A1 * A5)\big) \quad (2.37)$$

Semua data tersebut dihitung dengan cara baris dikalikan dengan kolom sehingga mendapatkan nilai *dot product* pada setiap tabel.

4. Hasil perhitungan kernel yang didapatkan akan dilanjutkan dengan menghitung matriks *hessian* dengan persamaan 2.38 sebagai berikut.

$$D_{ij} = y_i y_j\big(K(\vec{x}_i.\vec{x}_j) + \lambda^2\big) \quad (2.38)$$

dimana:

D_{ij} = elemen matriks hessian ke-ij

y_i = kelas data ke-i

y_j = kelas data ke-j

λ = batas teoritis yang akan diturunkan

5. Mencari nilai error dengan persamaan 2.39 sebagai berikut.

$$E_i = \sum_{j=1}^{l} \alpha_j D_{ij} \quad (2.39)$$

dimana:

E_i = nilai error data ke-i

D_{ij} = elemen matriks hessian ke-ij

6. Menghitung nilai δ_{α_i} (*delta alpha*) dengan persamaan 2.40 sebagai berikut.

$$\delta_{\alpha_i} = \min\{\max[\gamma(1 - E_i), -\alpha_i], C - a_i\} \quad (2.40)$$

7. Menghitung nilai δ_{α_i} (*delta alpha*) akan digunakan untuk menghitung nilai α baru dengan persamaan 2.41 sebagai berikut.

$$a_i = a_i + \delta_{\alpha_i} \quad (2.41)$$

8. Mencari nilai bias dengan cara menghitungnya menggunakan persamaan 2.42, 2.43, dan 2.44 sebagai berikut.

$$b = -\frac{1}{2}(\langle \vec{w}.\vec{x}_{-1} \rangle + \langle \vec{w}.\vec{x}_{+1} \rangle) \quad (2.42)$$

$$\langle \vec{w}.\vec{x}_{-1} \rangle = \sum \alpha_i . y_i . D_{ij} \quad (2.43)$$

dimana:

α_i = nilai *alpha* baru ke-i

y_i = target ke-i (kelas negatif)

D_{ij} = elemen matriks *hessian* ke-ij

$$\langle \vec{w}.\vec{x}_{+1} \rangle = \sum \alpha_i . y_i . D_{ij} \quad (2.44)$$

dimana:

α_i = nilai alpha baru ke-i

y_i = target ke-i (kelas positif)

D_{ij} = elemen matriks *hessian* ke-ij

Sebelum mencari nilai bias terlebih dahulu menghitung nilai w :

W_{i^+} = nilai data dengan *alpha* terbesar dikelas positif

W_{i^-} = nilai data dengan *alpha* terbesar dikelas negatif

9. Setelah mendapatkan nilai dari α, w dan b secara keseluruhan, nilai tersebut akan disimpan untuk dijakikan model untuk ke tahap berikutnya yaitu pengujian dataset. Tahap awal dari pengujian ini yang dilakukan yaitu menggunakan setiap fungsi perkalian untuk menghitung hasil *dot product* dari data uji dengan semua data pelatihan. Saat melakukan perhitungan nilai kernel, variabel x merupakan data uji yang akan digunakan dan variabel y merupakan

semua data latih yang sebelumnya digunakan. Setelah selesai melakukan perhitungan kernel dilanjutkan dengan melakukan perhitungan *decision function* (fungsi keputusan) dengan persamaan 2.45 atau 2.46 sebagai berikut.

$$f(x) = sign(w.x + b)\ atau \quad (2.45)$$

$$f(x) = sign\left(\sum_{i=1}^{m} \alpha_i y_i K(x_i.x_j) + b\right) \quad (2.46)$$

Semua data yang telah diuji akan dihitung akurasi keakuratannya sesuai dengan tingkat kebenaran yang dilakukan metode *support vector machine* dengan *confusion matrix* [59].

2.3.0 *Confusion Matrix*

Confusion matrix dapat dipahami sebagai alat yang berguna untuk menganalisis sejauh mana (*classifier*) dapat mengenali tupel dari kelas yang berbeda [31]. *Confusion Matrix* adalah array dua dimensi, satu dimensi diindeks oleh kelas objek yang sebenarnya dan dimensi lainnya diindeks oleh kelas yang ditentukan oleh (*classifier*). Kasus khusus (*confusion matrix*) sering digunakan dalam dua kelas (spesifikasi kelas positif). Dan lapisan negatif lainnya [32].

		Predicted class		
		Yes	No	Total
Actual class	Yes	TP	FN	P
	No	FP	TN	N
	Total	P`	N`	P + N

Gambar 2. 13 *Confusion matrix*, menampilkan total tupel *positive* dan *negative* (Han dan Kamber, 2011)

Keterangan :

- TP (*true positive*) : nilai asli positif & nilai prediksi positif
- FP (*false positive*) : nilai asli negatif & nilai prediksi positif
- FN (*false negative*) : nilai asli positif & nilai prediksi negatif
- TN (*true negative*) : nilai asli negatif & nilai prediksi negatif [24]

Ada beberapa rumus umum dapat digunakan untuk menghitung performansi dari algoritma yang digunakan. Adapun persamaan yang digunakan pada penelitian ini

yaitu akurasi, *recall*, presisi, f1-*score*, dan *error rate* yang dapat menampilkan dalam bentuk persentase [60].

a. Akurasi. Akurasi merupakan jumlah proporsi keakuratan prediksi yang benar. Adapun perhitungan akurasi dapat dilakukan dengan persamaan 2.47 berikut.

$$Akurasi = \frac{TP+TN}{TP+FP+TN+FN}\ x\ 100\% \text{ [31], [61]} \quad (2.47)$$

b. Recall. Recall adalah rasio prediksi yang benar-benar positif terhadap keseluruhan data dengan nilai positif yang sebenarnya mewakili tingkat keberhasilan model dalam mencari informasi. Adapun perhitungan *recall* dapat dilakukan dengan persamaan 2.48 berikut.

$$Recall = \frac{TP}{TP+FN} x100\% \quad (2.48)$$

c. Presisi. Presisi merupakan rasio prediksi yang bernilai benar positif yang dibandingkan dengan keseluruhan hasil yang diprediksi positif untuk menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Adapun perhitungan presisi dapat dilakukan dengan persamaan 2.49 berikut.

$$Presisi = \frac{TP}{TP+FP} x100\% \quad (2.49)$$

d. F1-s*core*. F1-s*core* merupakan hasil perbandingan rata-rata antara *recall* dan presisi yang dibobotkan. secara representasinya, jika f1-score memiliki skor yang baik berarti mengindikasikan bahwa model klasifikasi yang dibuat memiliki presisi dan recall yang baik. Adapun perhitungan *f1-score* dapat dilakukan dengan persamaan 2.50 berikut.

$$F1\ Score = 2\frac{(recall*presisi)}{(recall+presisi)} \quad (2.50)$$

e. Error rate. Error rate adalah proporsi kasus yang salah diidentifikasi dalam kaitannya dengan jumlah total kasus. Adapun perhitungan *error rate* dapat dilakukan dengan persamaan 2.51 berikut.

$$Error\ Rate = \frac{FP+FN}{Total\ Data} x100\% \quad (2.51)$$