BAB II

STUDI PUSTAKA

2.1 Tinjauan Pustaka

Penelitian tentang pendeteksi pemutuan pada buah manggis telah dilakukan sebelumnya, salah satunya oleh Sandra (2007) dalam penelitiannya yang berjudul "Pengembangan Pemutuan Buah Manggis untuk Ekspor Secara Non-Destruktif dengan Jaringan Syaraf Tiruan", tujuan dari penelitian ini untuk membangun sistem kecerdasan buatan berbasis jaringan syaraf tiruan, untuk mekanisme pemeriksaan dan pemutuan buah manggis segar secara non-destruktif dengan menggunakan metode JST. Parameter input yang digunakan adalah hasil dari pengolahan citra dengan perangkat penunjang *video capture* MATROX Meteor untuk pemutuan manggis bagian luar dan gelombang ultrasonik pada frekuensi 50kHz untuk pemutuan manggis dalam. Penelitian ini berfokus pada pemutuan buah manggis dan hasil penelitian menunjukkan ketepatan pemutuan berdasarkan diameter menghasilkan angka 94%, ketepatan pemutuan manggis dengan JST sebesar 95 % untuk kelas SNI dan 92 % untuk pendugaan rasio gula/asam. Namun peubah yang dipakai untuk penentuan kerusakan bagian buah manggis dengan teknik ultrasonik pada penelitian ini hanya kecepatan, sehingga perlu dikaji penggunaan parameter karasteristik gelombang ultrasonik lainnya.

Penelitian tentang identifikasi dan klasifikasi penyakit pada buah pernah dilakukan oleh Ranjit K.N. dkk. (2016), dalam penelitiannya yang berjudul "*Identification and Classification of Fruit Diseases*" bertujuan untuk mengidentifikasi dan mengklasifikasikan buah terhadap penyakit.

Penyakit pada buah mengurangi hasil panen dan juga menyebabkan memburuknya varietas hasil buah dari budidaya. Metode yang digunakan pada penelitian ini menggunakan pendekatan berbasis pengolahan citra yang terdiri dari beberapa langkah utama, seperti segmentasi menggunakan algoritma *K-means* dan *C-Meansclustering*, langkah performa dari algoritma segmentasi menggunakan metode *Measure of overlapping* (MOL), *Measure of under-segmentation* (MUS), *Measure of over segmentation* (MOS), *Dice similarity measure* (DSM), *Error-rate* (ER). Hasil penelitian menunjukkan bahwa mengklasifikasikan penyakit pada buah mengunakan segemntasi *K-means* memiliki akurasi yang relatif lebih tinggi dibanding menggunkan algoritma *C-Meansclustering*.

Penelitian pengolahan citra telah banyak dilakukan sebelumnya dengan berbagai macam metode pengolahan citra, salah satunya dengan menggunakan metode *curvelet*, penelitian menggunakan metode *curvelet* pernah dilakukan oleh Khoje, A. dkk., (2013) dalam penelitiannya "*Automated Skin Defect Identification System for Fruit Grading Based on Discrete Curvelet Transform*" tujuan dari penelitian ini untuk mengembangkan metodologi untuk menilai kualitas permukaan buah pada resolusi citra rendah dan tinggi menggunakan metode multiskala transformasi *curvelet*. Penelitian ini menggunakan 4 ekstraksi ciri sebagai masukan, keempat ciri ini tersebut yaitu rata-rata (*mean*), standar deviasi (*std*), *energy*, dan *entropy*. Keempat ekstraksi ciri tersebut kemudian saling dikombinasikan sebagai ekstraksi ciri. Metode yang digunakan untuk mengklasifikasikan kualitas permukaan buah adalah transformasi *curvelet* dengan klasifikasi *Support Vector Machine* (SVM) dan *Probabilistik Neural Networks* (PNN). Hasil penelitian menunjukkan bahwa klasifikasi SVM menghasilkan akurasi terbaik dengan nilai

96%. Studi ini menyimpulkan bahwa berdasarkan transformasi *curvelet* memberikan wawasan menjanjikan untuk memperkirakan kerusakan kulit buah.

Penelitian pengolahan citra menggunakan metode *wavelet* dilakukan oleh (Riyadi & dkk., 2008), pada penelitian ini bertujuan untuk mendeteksi buah pepaya cacat dan tidak cacat yang digunakan sebagai salah satu kriteria utama untuk menentukan kesesuaian buah pepaya yang akan diekspor. Penelitian ini menggunakan metode *wavelet* yang digunakan untuk menghasilkan ekstraksi ciri dari citra buah pepaya dan hasil ekstraksi ciri akan digunakan dalam proses klasifikasi buah dengan menggunakan metode *linear discriminant analysis* (LDA). Hasil pengujianyang dilakukan menghasilkan akurasi lebih dari 98%.

Penelitian tentang pengklasifikasian buah dengan pengolahan citra menggunakan metode *curvelet* pernah dilakukan oleh Marshalina, dkk., (2012) dengan objek buah mangga. Dalam penelitiannya yang berjudul "Klasifikasi Buah Mangga Berdasarkan Bentuk dan Warna dengan Metode *Curvelet*" bertujuan menciptakan suatu sistem berbasis *software* yang dapat mengidentifikasi jenis suatu mangga dengan mendeteksi citra mangga tersebut ke dalam sistem. Citra yang telah diakuisisi secara *offline* kemudian dijadikan sebagai citra latih dan citra uji yang selanjutnya akan diekstraksi cirinya dengan metode Transformasi *Curvelet*, serta dilakukan proses pengenalan dari ciri tersebut dengan metode *k-Nearest Neighbor* (k-NN). Dari hasil pengujian performansi sistem, maka diketahui bahwa performansi sistem mencapai akurasi tertinggi saat proses ekstraksi ciri menggunakan *Curvelet* skala 5 orientasi 16 dengan parameter klasifikasi yang diatur pada k-NN dengan akurasi sistem yang diperoleh tersebut mencapai 97% dengan waktu komputasi ± 11 detik.

Salah satu algoritma pembelajaran pengklasifikasian dalam pengolahan citra adalah *Support Vector Machine* (SVM). SVM dapat digunakan untuk memecahkan masalah klasifikasi data karena dapat memuat banyak fitur ekstraksi sekaligus. Penerapan metode SVM pernah dilakukan oleh Durgesh, S. K., & Lekha, B. (2009). dalam penelitiannya "*Data Classification using Support Vector Machine"* menunjukan bahwa pada *Support Vector Machine* dengan *kernel method"* menghasilkan klasifikasi yang lebih baik dibandingkan dengan jenis *Rule Based Classifier, K-NN Classifier, dan Local Transfer Function Classifier*. Ia juga menjelaskan bahwa algoritma *kernel Radial Basis Function* menghasilkan klasifikasi yang akurat dibandingkan *kernel polynomial* dan *sigmond*.

Penerapan pengklasifikasian menggunakan SVM juga pernah dilakukan oleh Djati Kerami & Hendri Murfi (2004), dalam penelitiannya yang berjudul "Kajian Kemampuan Generalisasi *Support Vector Machine* dalam pengenalan jenis *Spelice Sites* Pada Barisan DNA" ini metode yang digunakan pada penelitian adalah menggunakan matematis model SVM dalam memecahkan pengenalan pola, SVM mengetahui bentuk pola yang nantinya penelitian gunakan untuk kajian awal dan penentuan jenis *splice site* pada suatu barisan DNA. Hasil yang di peroleh pada penelitian ini menunjukan tingkat akurasi yang tinggi yaitu sebesar 95.4%, namun penulis hanya menggunakan metode klassifikasi SVM, kekurangan peneliti tidak menguji data lain untuk mendapat hasil yang lebih maksimal dan mengetahui hasil lain.

Penelitian tentang SVM juga dilakukan oleh Fitri Damayanti, Agus Zainal Arifin & Rully Soelaiman (2010). Dalam penelitianya yang berjudul "Pengenalan Citra Wajah Menggunakan Metode *Two-Dinemnsional Linear Discriminant*

Analysis Dan *Support Vektor Machine*" Penelitian ini akan mengembangkan aplikasi pengenalan wajah yang diintegrasikan dengan metode TDLDA dan SVM untuk pengenalan wajah. Dengan kombinasi kedua metode tersebut terbukti dapat memberikan hasil yang optimal dengan tingkat akurasi pengenalan antara 84,18% sampai 100% dengan uji coba menggunakan basis data ORL, YALE, dan BERN.

Dari beberapa tinjauan pustaka diatas, akan dilakukan riset tentang deteksi cacat pada permukaan buah manggis berbasis metode pengohan citra menggunakan metode *Support Vektor Machine* (SVM). Metode SVM digunakan untuk memecahkan masalah klasifikasi data karena dapat memuat banyak fitur ekstraksi sekaligus dan penggunaannya yang memiliki proses komputasi yang relatif cepat, dan deteksi cacat pada permukaan buah manggis berbasis pengolahan citra ini dapat membantu dalam meningkatkan kualitas buah manggis.

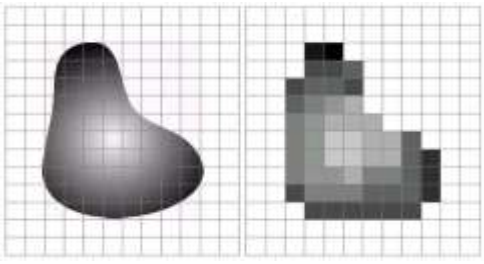
2.2 Landasan Teori

2.2.1 Deteksi Cacat

Deteksi cacat digunakan untuk mengetahui area cacat pada citra sehingga diketahui berapa presentase area cacatnya (Perwiranto, 2011). Cacat adalah kekurangan yang menyebabkan nilai atau kualitas dari suatu barang menjadi kurang baik atau kurang sempurna, cacat sangat memengaruhi kualitas dari suatu benda atau barang sehingga dalam hal ini kualitas telah menjadi salah satu faktor penting dalam pengambilan keputusan untuk suatu pemilihan.

2.2.2 Pengertian Citra Digital

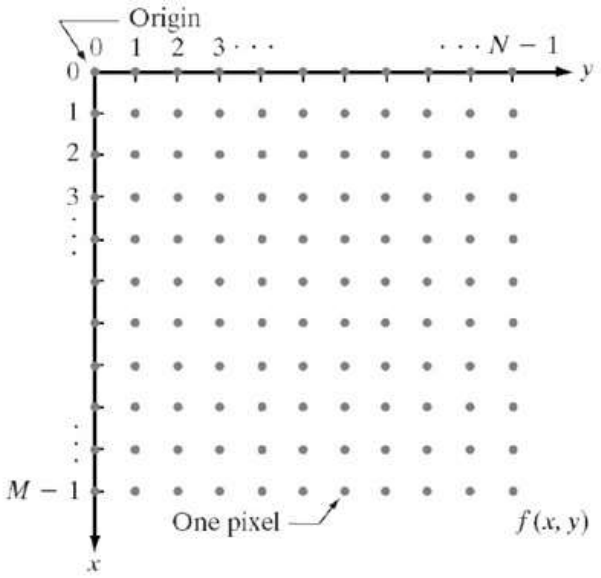
Citra adalah suatu gambaran, kemiripan, atau imitasi dari suatu objek yang dikeluarkan dari suatu alat perekam. Citra terbagi dua yaitu citra yang bersifat analog dan citra yang bersifat digital. Citra analog adalah citra yang bersifat kontinue seperti gambar pada monitor televisi, foto sinar X, hasil CT Scan dll., sedangkan pada citra digital adalah citra yang dapat diolah oleh komputer (Sutoyo et al. 2009).



Gambar 2.1 *Citra kontinu (kiri), Citra digital (kanan)*

Citra dapat didefinisikan sebagai sebuah fungsi dua dimensi, f (x, y) dimana x dan y merupakan koordinat bidang datar, dan harga fungsi f disetiap pasangan koordinat (x, y) disebut intensitas atau level keabuan (*grayscale*) dari gambar di titik itu. Jika x, y dan f semuanya berhingga (*finite*) dan nilainya diskrit, maka gambar itu disebut citra digital. Sebuah citra digital terdiri dari sejumlah elemen yang berhingga, dimana masing-masing mempunyai lokasi dan nilai tertentu. Sebuah citra digital dapat mewakili oleh sebuah matriks yang terdiri dari M kolom N baris, dimana perpotongan antara kolom dan baris disebut piksel (piksel = *picture element*), yaitu elemen terkecil dari sebuah citra. Piksel mempunyai dua parameter, yaitu koordinat dan intensitas piksel merupakan balok-balok bangunan dasar dari sebuah citra digital.

Citra digital dapat dimodelkan sebagai suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik atau koordinat pada citra tersebut dan elemen matriksnya menyatakan tingkat keabuan pada titik tersebut.



Gambar 2.2 *Koordinat citra digital*

Pemodelan citra digital dalam bentuk matriks berukuran N x M, sebagai berikut :

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

Gambar 2.3 *Representasi citra digital dalam matriks*

Sebuah piksel merupakan warna atau nilai kecemerlangan yang menempati sebuah tempat spesifik pada sebuah citra. Sebuah citra seperti sebuah *grid* dengan masing-masing kotak persegi di dalam *grid* berisi satu warna atau piksel. Sebuah citra 8 dengan resolusi 1024x768 adalah sebuah grid dengan 1024 kolom dan 768 baris, yang mana berisi 1024x768 = 786432 piksel. Banyaknya piksel pada sebuah citra tidak menunjukkan dimensi fisik dari sebuah citra. Dengan kata lain, satu piksel tidak sama dengan satu *millimeter*, satu *micrometer* atau satu *nanometer*. Seberapa "luas" sebuah piksel akan tergantung pada pengaturan piksel per inch (PPI) untuk citra tersebut.atau warna.

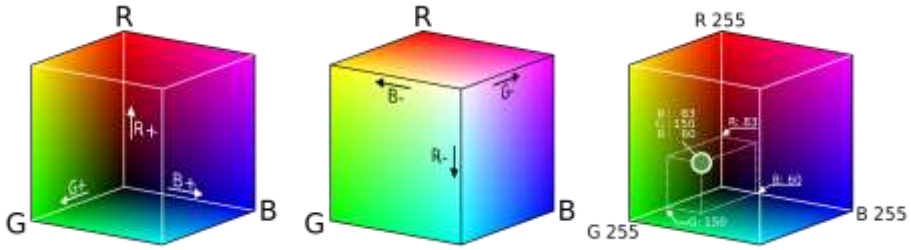
2.2.3 Pengertian Pengolahan Citra Digital

Pengolahan citra adalah sebuah proses pengolahan yang inputnya adalah citra digital dan outputnya dapat berupa citra atau sekumpulan karasteristik atau parameter yang berhubungan dengan citra. Pengolahan citra merupakan istilah untuk bermacam-macam teknik gambar berdimensi dua yang dapat diolah dengan mudah (Efford, 2000). Pengolahan citra digital adalah sebuah disiplin ilmu yang mempelajari hal-hal yang berkaitan dengan perbaikan kualitas gambar (peningkatan kontras, transformasi geometrik), melakukan pemilihan citra ciri (*feature image*) yang optimal untuk tujuan analisis, melakukan proses penarikan informasi atau deskripsi objek atau pengenalan objek yang terkandung dalam citra, melakukan kompresi atau reduksi data untuk tujuan penyimpanan data, transmisi data, dan waktu proses data. Input dari pengolahan citra adalah citra, sedangkan *output*-nya adalah citra hasil pengolahan (Sutoyo et al, 2009).

2.2.4 Citra Warna (RGB)

Citra RGB adalah citra yang menggunakan 3 kanal warna yaitu merah, hijau dan biru. Selain pada warna tersebut merupakan suatu penumpukan dari matrik, yang mana masing-masing matrik mempersentasikannilai warna merah, hijau, dan biru disetiap piksel, dimana setiap piksel berkaitan dengan tigs nilai. (Sianipar, 2013).

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi dari tiga warna dasar (*RGB = Red Green Blue*). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 *byte*, yang berarti setiap warnanya mempunyai gradasi sebanyak 255 warna dan pada komputer bilangan yang digunakan bilangan biner 8 digit, sehingga skala yang digunakan adalah 256. Berarti setiap piksel mempunyai kombinasi warna sebanyak 2^8 x 2^8 x 2^8 = 2^{24} = 16 juta warna lebih. Dalam matematika sebuah jenis warna dapat dibayangkan dengan *vector* dimensi tiga komponen yaitu komponen x y dan z. Misalkan *vector* yang ditulis r = (x, y, z) untuk warna, komponen tersebut digantikan oleh komponen Merah (*Red*), Hijau (*Green*) dan Biru (*Blue*). Jadi, sebuah jenis warna dapat dituliskan sebagai berikut: RGB (30, 75, 255), warna putih = RGB (255, 255, 255), sedangkan warna hitam = RGB (0, 0, 0). Bentuk representasi warna dari sebuah citra pada Gambar 2.4



Gambar 2.4 *Citra RGB*

2.2.5 Citra Abu–abu (*Grayscale*)

Citra *grayscale* merupakan citra digital yang hanya memiliki satu nilai kanal pada setiap pikselnya, dengan kata lain bagian *red = green = blue*. Nilai tersebut digunakan untuk menunjukkan tingkat intensitas. Warna yang dimiliki adalah warna dari hitam, keabuan dan putih. Tingkatan keabuan di sini merupakan warna abu dengan berbagai tingkatan dari hitam hingga mendekati putih (Darma, 2010).

Pada citra *grayscale* warna hitam pada bagian ini memiliki nilai terendah, dan nilai warna putih menjadi nilai terkuat, atau nilai warna putih lebih tinggi dari nilai warna hitam. Citra *grayscale* berbeda dengan citra biner atau citra "hitam-putih". Apabila citra hitam putih hanya mengenal dua warna yaitu "hitam" dan warna "putih" sedangkan *grayscale* mempunyai variasi yang banyak, karena ada nilai-nilai diantara nilai minimum (biasanya = 0) dan nilai maksimum. Banyaknya kemungkinan nilai minimum dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Warna dari sebuah citra *grayscale* dapat dilihat pada Gambar 2.5

Gambar 2.5 *Intensitas citra grayscale, hitam = 0 dan putih = 256*

Format citra ini disebut skala keabuan, karena pada umumnya warna yang dipakai adalah antara warna hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya, sehingga warna diantara warna hitam dan warna putih adalah warna abu-abu. Citra *grayscale* disimpan dalam format 8bit untuk setiap pikselnya yang memungkinkan sebanyak 256 intensitas. Untuk mengubah citra berwarna

(RGB) menjadi citra *grayscale* dapat dilakukan dengan menghitung rata-rata dari nilai R, G, dan B sehingga dapat dilihat pada rumus berikut:

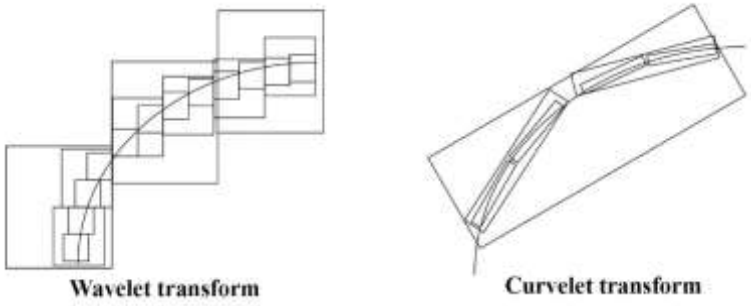
$$f_0(x,y) = \frac{f_i^R(x,y) + f_i^G(x,y) + f_i^B(x,y)}{3} \quad \text{.....(2.1)}$$

2.2.6 Curvelet

Teori *curvelet* merupakan suatu konsep yang relatif baru dikembangkan. Teori *curvelet* pertama kali diajukan dan dikembangkan oleh E. J. Candes dan D. L. Donoho pada tahun 2000. Teori *curvelet* adalah representasi dari gambaran kurva (*curve*) sebagai superposisi dari fungsi yang memberikan berbagai macam nilai panjang dan lebar yang berdasarkan dari hukum skala (*scaling law*) *width = length*^{2}. (Starck et al, 2002) .

Curvelet sebagai metode tranformasi yang merepresentasi fungsi kurva (*curve*) muncul akibat dari metode transformasi *wavelet* yang kurang efesien karena fungsi linier dapat mengakibatkan pembusukan gambar (*decompose image*) dengan cara *isotropic*. Pada transormasi *wavelet* untuk dapat berjalan dengan baik dibutuhkan lebih banyak koefisien dari gambar, lebih banyak *level* dari *decomposition*s pada gambar yang mengakibatkan dibutuhkan waktu yang lama. Dalam metode *curvelet,* fungsi utama dari kurva (*curve*) yang dihasilkan dari persamaan skala (*scaling law*) *width = length*^{2}. Metode ini dapat berjalan dengan multiskala transformasi yang beroperasi pada gambar dengan cara a*nisotropic*, hal ini memberikan metode terbaik dengan mengurasi koefisien *curvelet* dan mengurangi waktu yang dibutuhkan dibanding dengan transformasi *wavelet*.

(Jianwei dan G. Plonka, 2010), perbandingan transformasi *wavelet* dan *curvelet* dapat dilihat pada Gambar 2.7 sebagai berikut:



Gambar 2.6 *Perbandingan transformasi wavelet (kiri) dan curvelet (kanan)*

2.2.7 Transformasi *Curvelet*

Transformasi *curvelet* adalah transformasi skala jamak (*multiscale*) terarah yang mengijinkan penampilan sparsa *nonadaptive* hampir optimal dari suatu obyek yang mempunyai banyak tepi. Transformasi *Curvelet* diketemukan dalam riset untuk tujuan mengatasi keterbatasan transformasi *wavelet* tahun 1999 oleh Candes dan Donoho. Sebenarnya, inti dari transformasi *curvelet* adalah transformasi *ridgelet*. Dalam tahun 1999, transformasi *wavelet* yang secara geometrik *isotrop* dikembangkan menjadi *anisotrop* dan disebut transformasi *ridgelet* (ridgelet *transform*) yang diperkenalkan oleh Candes dan Donoho. Transformasi *ridgelet* optimal ketika menampilkan singularitas garis lurus namun singularitas garis lurus tersebut tidak dapat ditampilkan dalam aplikasi real. Untuk itu dikembangkan analisis tentang suatu garis atau kurva, ide dasarnya adalah membuat partisi kemudian ditransformasi dengan ridglet untuk mendapat citra bagian.

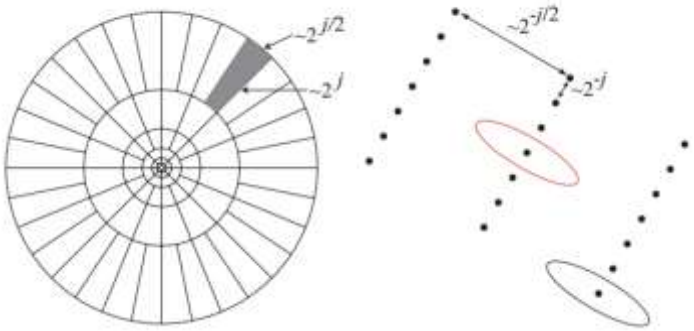
Pembagian blok-blok transformasi tersebut kemudian dikenal dengan transformasi *curvelet* generasi pertama, namun aplikasinya masih terbatas karena geometrinya belum jelas. Transformasi *curvelet* generasi kedua lebih sederhana, berdasarkan teknik pembagian rekuensi diperkenalkan kemudian. Transformasi *curvelet* generation kedua telah mempertunjukkan sebagai suatu alat yang sangat efisien untuk berbagai macam aplikasi dalam pengolahan citra.

Transformasi *curvelet* sebelumnya memiliki dua bentuk antara lain *Continuous Curvelet Transforms* dan *Discrete Curvelet Transform*. *Continuous Curvelet Transforms* melakukan transformasi dengan cara membagi citra pada domain frekuensi sepanjang sudut radial melingkar menggunakan bagian-bagian jendela, sedangkan *Discrete Curvelet Transform* melakukan transformasi lanjutan dengan cara membagi gambar menggunakan kotak tengah yang ada pada gambar. Tranformasi *curvelet* akan menghasilkan informasi yang berlebih (*redundant information*) sehingga dapat memrepresentasikan sinyal yang berada pada tepian kurva gambar sehingga tranformasi *curvelet* didesain ulang kemudian dan diperkenalkan sebagai *Fast Discrete Curvelet Transform* (FDCT) (Candes, Demanet, & Donoho, 2005).

Generasi pertama dari transformasi *curvelet* menggunakan serangkaian langkah yang komplek yang melibatkan *ridgelet analysis* dalam melakukan transformasi pada gambar, hal tersebut menyebabkan kinerja transformasi menjadi sangat lambat. Pada generasi kedua pada transformasi *curvelet* cenderung lebih mudah dipahami dan digunakan dikarenakan membuang bagian pada penggunaan *ridgelet transform,* sehingga mengurangi jumlah redudansi dalam proses

transformasi dan meningkatkan kecepatan secara signifikan dibanding dengan generasi pertama (Candes, Demanet, & Donoho, 2005).

Untuk mengimplementasikan transformasi *curvelet*, langkah pertama yang diambil adalah dengan cara mengambil gambar 2D dengan menggunakan 2D *Fast Fourier Transform* (FFT), kemudian bidang frekuensi gambar dibagi menjadi irisan-irisan seperti Gambar 2.8



Gambar 2.7 *Irisan transformasi Curvelet (kiri) dan spasial domain (kanan)*

Bentuk parabola yang tersusun atas irisan-irisan pada gambar adalah hasil dari pembagian *fourier plane* menjadi *radial* (lingkaran konsentris) dan pecahan-pecahan dari sudut lingkaran. Lingkaran konsentris bertanggung jawab atas penguraian gambar menjadi beberapa skala dan perpecahan sudut yang membagi pecahan-pecahan melewati citra ke sudut pandang atau orientasi yang berbeda, sehingga untuk menghadapi irisan tertentu membutuhkan definisi skala *j* dan *angle* θ. Dari digital transformasi ini menghasikan garis linier dan *input Array Cartesian* dari bentuk $[t1, t2], 0 \leq t1, t2 < n$, yang memungkinkan menghasilkan *output* sebagai sekumpulan koefisien cD(j, l, k), dimana masing-masing cD (j, l, k) adalah bentuk gelombang *curvelet* digital yang diperoleh dari persamaan berikut:

$$C^D(j,l\,k) = \sum_{0 \le t1,t2<n} f\,[t1,t2]\overline{\phi^D_{j,l,k}\,[t1,t2]} \quad \text{.....(2.2)}$$

2.2.8 Ekstraksi Ciri

Ekstraksi ciri merupakan suatu proses pengambilan ciri dari suatu bentuk yang nantinya nilai yang didapatkan akan dianalisis dan digunakan sebagai bahan analisis. Untuk mendapatkan nilai dari suatu ciri dilakukan dengan cara menghitung jumlah titik atau piksel yang ditemui dalam setiap pengecekan, dimana pengecekan dilakukan dalam berbagai arah pada koordinat kartesian dari citra digital yang dianalisis, yaitu vertikal, horizontal, diagonal kiri, dan diagonal kanan.

Ekstraksi ciri yang digunakan adalah nilai rata-rata(*mean*), standar deviasi, *energy* dan *entropy* yang dihasilkan dari perhitungan matriks koefisien curvelet dari masing-masing citra sehingga nantinya ekstraksi ciri yang dipilih ini akan digunakan untuk menyelidiki dan memberikan penilaian dari kualitas buah manggis. Empat ekstraksi ciri seperti mean, standar deviasi, *energy* dan *entropy* menunjukkan kekuatan diskriminasi yang tinggi dalam pemeriksaan kualitas buah (Khoje, S. A., dkk. 2013).

Perhitungan nilai ekstraksi ciri citra menggunakan rumus sebagai berikut:

1. **Rata-rata**

   Persamaan rata – rata :

$$\boldsymbol{\mu} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{X_1} \text{.....(2.3)}$$

   Nilai rata-rata dihitung dengan cara menjumlahkan nilai setiap piksel mulai dari piksel ke-1 hingga piksel ke-N kemudian dibagi dengan jumlah piksel yang ada.

2. Standar Deviasi

Persamaan standar deviasi:

$$\boldsymbol{\sigma} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{x_i} - \vec{\boldsymbol{x}}_{\boldsymbol{l}})^{\mathbf{2}}} \quad \text{.....(2.4)}$$

Nilai standar deviasi dihitung dengan cara mengurangkan nilai setiap piksel ke-1 sampai ke-N dengan nilai rata-rata. Hasil pengurangan ini dikuadratkan dan dijumlahkan pada setiap piksel ke-1 sampai ke-N. Hasil penjumlahan ini kemudian diakar kuadratkan untuk mendapatkan nilai standar deviasi.

3. *Entropy*

Entropy berfungsi dalam menunjukkan ukuran ketidakteraturan dari sebuah bentuk. Nilai *H* akan bernilai besar jika menggunakan citra dengan transisi derajat keabuan yang merata dan bernilai kecil jika struktur citra yang digunakan tidak teratur (bervariasi). Nilai *entropy* menunjukkan keteracakan distribusi derajat keabuan suatu citra. Semakin acak distribusi derajat keabuannya, semakin tinggi nilai *entropy* yang dihasilkan. Persamaan untuk menghitung *entropy* dapat dilihat pada pada persamaan berikut:

$$H = -\sum_{i=1}^{G} p(di).log_2\, p(d_i) \quad \text{.....(2.5)}$$

4. *Energy*

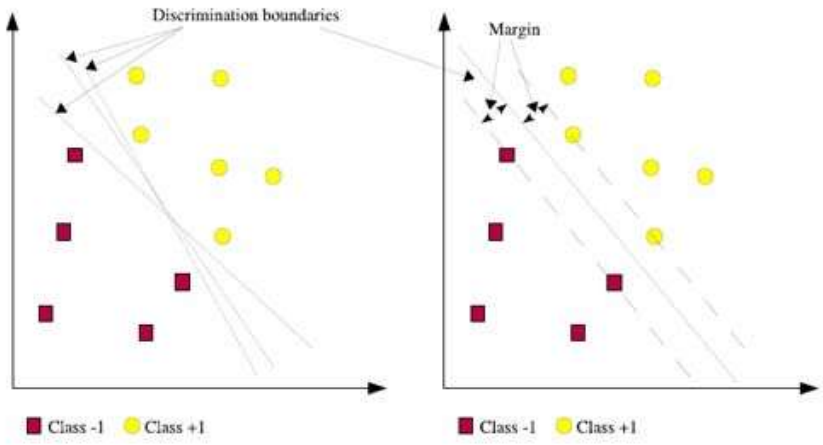
Energy adalah suatu fitur yang digunakan untuk mengukur konsentrasi pasangan intensitas pada sebuah matrik *co-accurrence*. Nilai *energy* akan menghasilkan nilai yang besar jika distribusi *level gray* citra mempunyai bentuk yang konstan atau periodik. Semakin tinggi nilai *entropy* maka nilai *energy* akan

semakin rendah. Hal ini dikarenakan, nilai *energy* menggambarkan keteraturan penyebaran derajat keabuan suatu citra, sehingga bisa dikatakan *energy* adalah *inverse* dari *entropy*. Persamaan untuk menghitung *Energy* dapat dilihat pada pada persamaan berikut:

$$E_k = \frac{1}{M\ x\ N} \sum_{i=1}^{M} \sum_{j=1}^{N} |x_k(i,j)| \quad \text{.....(2.6)}$$

2.2.9 Support Vector Machine (SVM)

Klasifikasi data adalah suatu proses untuk mengelompokan sejumlah data ke dalam grup golongan tertentu berdasarkan property atau nilai data tersebut. Salah satu metode klasifikasi data adalah *Support Vector Machine*. Menurut Santoso (2007) *Support Vector Machine* (SVM) adalah suatu teknik untuk melakukan prediksi, baik dalam kasus klasifikasi maupun regresi.



Gambar 2.8 *Konsep Support Vector Machine (SVM)*

Menurut Santoso (2007) konsep kerja SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Gambar 2.9 memperlihatkan beberapa

pattern yang merupakan anggota dari dua buah *class* : +1 dan -1. *Pattern* yang tergabung pada *class* -1 disimbolkan dengan warna merah (kotak), sedangkan *pattern* pada *class* +1 disimbolkan dengan warna kuning (lingkaran).

Permasalahan klasifikasi dapat diterjemahkan dengan usaha menemukan garis (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah (*discrimination boundaries*) ditunjukkan pada Gambar 2.9. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur *margin hyperplane* tersebut, dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* dengan *pattern* terdekat dari masing-masing *class*. *Pattern* yang paling dekat inilah yang disebut dengan *supportvector*.

Garis solid pada gambar 2.9 sebelah kanan menunjukkan *hyperplane* yang terbaik, yaitu garis yang terletak tepat pada tengah-tengah kedua *class*, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM.

Data yang tersedia dinotasikan sebagai $\vec{x}_i \in \Re^d$ sedangkan label masing-masing dinotasikan $y_i \in \{-1, +1\}$ untuk $i = 1, 2, \ldots, l$ yang mana l adalah banyaknya data. Diasumsikan kedua *class* -1 dan +1 dapat terpisah secara sempurna oleh *hyperplane* berdimensi *d* , yang didefinisikan

$$\vec{w}.\vec{x} + b = 0 \quad \text{.....(2.7)}$$

Pattern $\vec{x}_i$ yang termasuk *class* -1 (*sampel negative*) dapat dirumuskan sebagai *pattern* yang memenuhi pertidaksamaan

$$\vec{w}.\vec{x}_i + b \leq -1 \text{.....(2.8)}$$

Sedangkan *pattern* $\vec{x}_i$ yang termasuk *class* +1 (*sample positif*) dirumuskan

$$\vec{w}.\vec{x}_i + b \geq +1 \text{.....(2.9)}$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya, yaitu $1/\|\vec{w}\|$. Hal ini dapat dirumuskan sebagai *Quadratic Programming* (QP) problem, yaitu mencari titik minimal persamaan (2.10), dengan memperhatikan *constraint* persamaan (2.11).

$$\min_{\vec{w}} \tau(w) = \frac{1}{2} \|\vec{w}\|^2 \text{.....(2.10)}$$

$$y_i \, (\vec{x}_i . \vec{w} + b) - 1 \geq 0 \; \forall i \text{.....(2.11)}$$

Problem ini dapat dipecahkan dengan berbagai teknik komputasi, di antaranya *Lagrange Multiplier*.

$$(L (\vec{w}, b, a) = \frac{1}{2} \|\vec{w}\|^2 - \sum_{i=1}^{l} a_i (y_i \, ((\vec{x}_i . \vec{w} + b) - 1)) \text{ dengan } i = 1, 2, ...,l) \text{....(2.12)}$$

α_i adalah *Lagrange multipliers*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan (2.12) dapat dihitung dengan meminimalkan *L* terhadap $\vec{w}$ dan *b*, dan memaksimalkan *L* terhadap α_i. Dengan memperhatikan sifat bahwa pada titik optimal gradient *L*=0, persamaan (2.12) dapat dimodifikasi sebagai maksimalisasi problem yang hanya mengandung saja α_i , sebagaimana persamaan (2.13)

$$\sum_{i=1}^{l} a_i - \frac{1}{2} \sum_{i,j=1}^{l} a_i a_j y_i y_j \vec{x}_i \vec{x}_j \text{.....(2.13)}$$

$$a_i \geq 0 \; (i = 1, 2, ...,l) \sum_{i,=1}^{l} a_i y_i = 0 \text{ .....(2.14)}$$

Dari hasil dari perhitungan ini diperoleh α_i yang kebanyakan bernilai positif. Data yang berkorelasi dengan α_i yang positif inilah yang disebut sebagai *support vector*. Penjelasan diatas berdasarkan asumsi bahwa kedua belah *class* dapat terpisah secara sempurna oleh *hyperlane*. Akan tetapi, umumnya dua buah *class* pada *input space* tidak dapat terpisah secara sempurna. Hal ini menyebabkan *constraint* pada persamaan (2.11) tidak dapat terpenuhi, sehingga optimisasi tidak dapat dilakukan. Untuk mengatasi masalah ini, SVM dirumuskan ulang dengan memperkenalkan teknik *soft margin*. Dalam *soft margin* persamaan (2.11) dimodifikasi dengan memasukan *slack variable* ξ_i ($\xi_i > 0$) sebagai berikut.

$y_i (\vec{x}_i . \vec{w} + b) + 1 \geq 0 \; \forall i$ ... ...(2.15)

Dengan demikian persamaan (2.4) diubah menjadi :

$\min_{\vec{w}} \tau(w, \quad \xi) = \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^{l} \xi i$ ... ...(2.16)

Parameter *C* dipilih untuk mengontrol *tradeoff* antara *margin* dan *error* klassifikasi ξ . Nilai *C* yang besar berarti akan memberikan penalti yang lebih besar terhadap *error* klasifikasi tersebut.

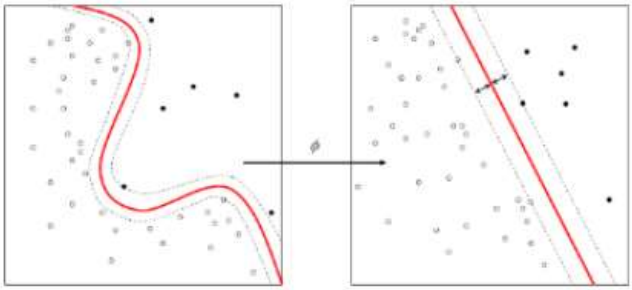
Pada umumnya masalah dalam domain dunia nyata (*real world problem*) jarang yang bersifat *linear separable*. Kebanyakan bersifat *non linear*. Untuk menyelesaikan problem *non linear*, SVM dimodifikasi dengan memasukkan fungsi *Kernel*. Dalam *non linear* SVM, pertama-tama data *x* dipetakan oleh fungsi F ke ruang vektor yang berdimensi lebih tinggi. Pada ruang vektor yang baru ini, *hyperplane* yang memisahkan kedua *class* tersebut dapat dikonstruksikan. Hal ini

sejalan dengan teori Cover yang menyatakan *"Jika suatu transformasi bersifat non linear dan dimensi dari feature space cukup tinggi, maka data pada input space dapat dipetakan ke feature space yang baru, dimana pattern-pattern tersebut pada probabilitas tinggi dapat dipisahkan secara linear"*.

Pemetaan ini dilakukan dengan menjaga topologi data, dalam artian dua data yang berjarak dekat pada *input space* akan berjarak dekat juga pada *feature space*, sebaliknya dua data yang berjarak jauh pada *input space* akan juga berjarak jauh pada *feature space*. Selanjutnya proses pembelajaran pada SVM dalam menemukan titik-titik *support vector*, hanya bergantung pada *dot product* dari data yang sudah ditransformasikan pada ruang baru yang berdimensi lebih tinggi. Karena umumnya transformasi F ini tidak diketahui, dan sangat sulit untuk difahami secara mudah, maka perhitungan *dot product* tersebut sesuai teori Mercer dapat digantikan dengan fungsi kernel yang mendefinisikan secara implisit transformasi Hal ini disebut sebagai *Kernel Trick* yang dirumuskan *Kernel trick* memberikan berbagai kemudahan, karena dalam proses pembelajaran SVM, untuk menentukan *support vector*, kita hanya cukup mengetahui fungsi kernel yang dipakai, dan tidak perlu mengetahui wujud dari fungsi non linear F . Berbagai jenis fungsi kernel dikenal.

Salah satu *Kernel Trick* yang banyak digunakan adalah *Radial Basis Function* (RBF). Menurut penelitian durgesh (2009), RBF merupakan *kernel trick* yang lebih baik dibandingkan *Polynomial* dan *Sigmond*. RBF dapat memetakan data sebagai titik pada dimensi yang lebih tinggi, memeliki sedikit parameter sehingga *tuning* lebih mudah, dan proses komputasi dan metode numerik yang lebih mudah.

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2), \ \gamma > 0$$

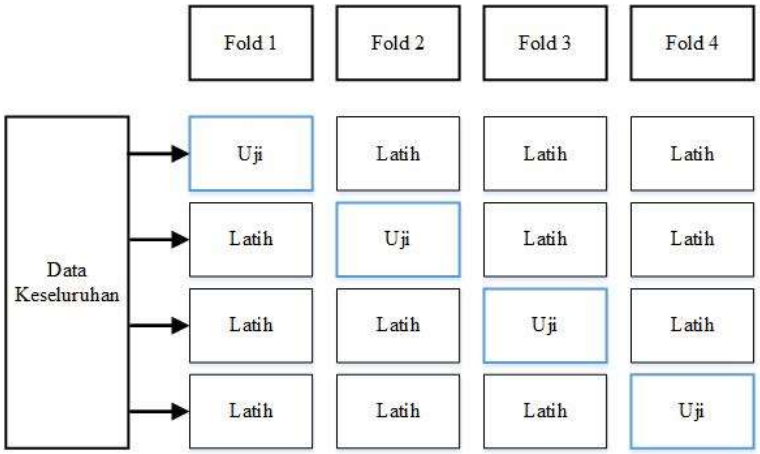


Gambar 2.9 *Hasil Kernel RBF*

2.2.10 K-Fold Cross Validation

Cross Validation merupakan salah satu teknik untuk menilai/memvalidasi keakuratan sebuah model yang dibangun berdasarkan dataset tertentu. Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang belum pernah muncul di dalam dataset. Data yang digunakan dalam proses pembangunan data model disebut data latih/*training*, sedangkan data yang akan digunakan untuk validasi model disebut dengan data uhi/*testing*. Salah satu metode *cross-validation* yang populer adalah *K-Fold Cross Validation*. Dalam teknik ini dataset dibagi menjadi sejumlah K-buah partisi secara acak. Kemudian dilakukan sejumlah K-kali eksperimen, dimana masing-masing eksperimen menggunakan data partisi ke-K sebagai data testing dan memanfaatkan sisa partisi lainnya sebagai data training (Muafiq, 2016).

Sebagai gambaran 4-*Fold Cross Validation* ditunjukan pada gambar 2.11. Untuk mendapatkan nilai akurasi ataupun ukuran penilaian lainnya dari hasil eksperimen yang dilakukan, dapat diambil nilai rataan dari seluruh eksperimen tersebut.



Gambar 2.10 *Data set pada K-Fold Cross Validation*