SKRIPSI

DETEKSI KARBON NANO-DOT (C-DOTS) DALAM CITRA SEM MENGGUNAKAN *SUPPORT VECTOR MACHINE* (SVM)

INDRA SETIAWAN

H021 18 1329

DEPARTEMEN FISIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2023

DETEKSI KARBON NANO-DOT (C-DOTS) DALAM CITRA SEM MENGGUNAKAN *SUPPORT VECTOR MACHINE* (SVM)

SKRIPSI

Diajukan Sebagai Salah Satu
Memperoleh Gelar Sarjana Sains
Pada Program Studi Fisika Departemen Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin

INDRA SETIAWAN

H021181329

DEPARTEMEN FISIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2023

HALAMAN PENGESAHAN

DETEKSI KARBON NANO-DOT (C-DOTS) DALAM CITRA SEM MENGGUNAKAN *SUPPORT VECTOR MACHINE* (SVM)

Disusun dan diajukan oleh:

INDRA SETIAWAN

H021181329

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

Pada 07 Juni 2023

Dan dinyatakan telah memenuhi syarat kelulusan.

Menyetujui,

Pembimbing Utama	Pembimbing Pertama
Eko Juarlin, S.Si, M.Si	Heryanto, S.Si, M.Si
NIP. 19811106 200812 1 002	NIP. 19911129 202005 3 001

Ketua Program Studi,

Prof. Dr. Arifin, M.T

NIP. 19670520 199403 1 002

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : INDRA SETIAWAN

NIM : H021181329

Program Studi : FISIKA

Jenjang : S1

Menyatakan dengan sebenarnya bahwa karya tulisan saya yang berjudul:

DETEKSI KARBON NANO-DOT (C-DOTS) DALAM CITRA SEM MENGGUNAKAN *SUPPORT VECTOR MACHINE* (SVM)

Adalah karya tulis berdasarkan hasil pemikiran dan penelitian saya, bukan merupakan hasil pengambil alihan tulisan maupun pemikiran orang lain. Jika terdapat karya orang lain dalam skripsi ini, maka akan dicantumkan sumber yang benar dan jelas. Demikian surat pernyataan ini saya buat dengan sebenarnya, jika dikemudian hari terdapat ketidakbenaran dan penyimpangan dalam pernyataan ini, maka saya berhak menerima sanksi atas perbuatan tersebut.

Makassar, 07 Juni 2023

Menyatakan

Indra Setiawan

H021181329

KATA PENGANTAR

Alhamdulilahi Rabiil'alamiin puji dan syukur penulis panjat kan kepada Allah SWT atas limpahan rahmat dan hidayah nya, berupa nikmat kesehatan dan kesempatan sehingga penulis berhasil menyelesaikan penulisan skripsi dengan judul "**Deteksi karbon Nano-dot (C-Dots) Dalam Citra SEM Menggunakan *Support Vector Machine* (SVM)**". Berbagai upaya telah dilakukan penulis untuk menyelesaikan penulisan skripsi ini sebagai salah satu syarat untuk menyelesaikan studi dan memperoleh gelar sarjana di program studi Fisika, Fakultas matematika dan ilmu pengetahuan alam Universitas Hasanuddin.

Penulis menyadari bahwa dalam penulisan skripsi ini banyak kesulitan dan hambatan yang dihadapi dan tidak terlepas dari dukungan berbagai pihak sehingga penulisan skripsi ini masih jauh dari kata sempurna. Namun atas kehendak nya hambatan tersebut berhasil dilalui oleh penulis sehingga penyusunan skripsi ini dapat diselesaikan. Oleh karena itu, dengan segala kerendahan hati, penulis mengucapkan banyak terimakasih kepada:

1. Bapak **Eko Juarlin, S.Si,M.Si** selaku pembimbing utama dan Bapak **Heryanto, S.Si,M.Si** selaku pembimbing pertama yang telah meluangkan banyak waktu dan tenaga nya dalam memberikan bimbingan, memberikan arahan, memberikan dukungan serta motivasi dan kepercayaan kepada penulis untuk berpikir kritis dan logis dalam melaksanakan penelitian dan penyusunan skripsi.
2. **Prof. Dr. rer-net Wira Bahari Nurdin** dan Bapak **Bannu, S.Si, M.Si.** selaku dosen penguji yang telah banyak meluangkan waktu dan tenaga nya untuk memberikan masukan dan saran yang membangun terkait penelitian dan kesempurnaan skripsi ini.
3. **Prof. Dr. Ir. Jamaluddin jompa, M,Sc.** selaku rektor Universitas Hasanuddin.
4. Bapak **Dr. Eng. Amiruddin** selaku Dekan Fakultas Matematika Dan Ilmu Pengetahuan Alam Universitas Hasanuddin.
5. **Prof. Dr. Erifin, M,T** selaku kepala Program Studi Fisika Fakultas

Matematika Dan Ilmu Pengetahuan Alam Univertsitas Hasanuddin.

6. **Seluruh dosen** Fakultas Matematika dan Ilmu Pengetahuan Alam yang telah membagikan ilmu nya dan membimbing penulis selama masa studi untuk memperoleh pengetahuan yang luas dan bermanfaat kepada penulis.
7. **Seluruh staff** departemen dan fakultas atas segala fasilitas dan pelayanan yang diberikan kepada penulis selama menempuh studi hingga penelitian dan penyusunan skripsi ini.
8. Kepada kedua orang tua tercinta ayahanda **Torisman** dan ibunda tersayang **Fatmawati** yang telah membimbing penulis dari kecil, tidak pernah lelah merawat dan memberikan dukungan kepada penulis baik secara mental maupun material. Semoga Ayah dan Ibu diberikan nikmat dan kesehatan yang panjang, Amiin.

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

ABSTRAK

Penelitian ini membahas penggunaan Support Vector Machine (SVM) untuk mendeteksi Karbon Nano-Dot (C-Dots) dalam citra SEM (*Scanning Electron Microscopy*). C-Dots adalah nanopartikel karbon dengan ukuran sangat kecil dan memiliki potensi aplikasi yang luas dalam bidang nanoteknologi. Metode yang diusulkan dalam penelitian ini mengintegrasikan SVM dengan teknik pemrosesan citra untuk mengidentifikasi dan mendeteksi C-Dots dalam citra SEM. Proses deteksi C-Dots melibatkan beberapa tahap, termasuk pra-pemrosesan citra, ekstraksi fitur, dan klasifikasi dengan SVM. SVM adalah metode pembelajaran mesin yang mampu memisahkan kelas data yang kompleks dengan membangun sebuah hyperplane yang optimal di antara kelas-kelas tersebut. Dalam penelitian ini, SVM dilatih menggunakan contoh citra yang telah diberi label sebagai C-Dots atau bukan C-Dots. Setelah pelatihan, SVM digunakan untuk mengklasifikasikan citra-citra baru dan mengidentifikasi apakah terdapat C-Dots di dalamnya. Hasil eksperimen menunjukkan bahwa metode yang diusulkan dapat mendeteksi C-Dots dengan akurasi yang baik. Dalam banyak kasus, metode ini berhasil mengenali C-Dots dengan tingkat akurasi di atas 90%. Hasil ini menunjukkan bahwa SVM berpotensi menjadi alat yang efektif dalam deteksi C-Dots dalam citra SEM.

Kata Kunci: *C-Dots, Bukan C-Dots , SEM, SVM.*

ABSTRACT

This study discusses the use of Support Vector Machine (SVM) to detect Carbon Nano-Dots (C-Dots) in SEM (Scanning Electron Microscopy) images. C-Dots are carbon nanoparticles with a very small size and have broad potential applications in the field of nanotechnology. The method proposed in this study integrates SVM with image processing techniques to identify and detect C-Dots in SEM images. The C-Dots detection process involves several stages, including image pre-processing, feature extraction, and classification with SVM. SVM is a machine learning method that is able to separate complex data classes by building an optimal hyperplane between these classes. In this study, SVM was trained using sample images that have been labeled as C-Dots or not C-Dots. After training, SVM is used to classify new images and identify whether there are C-Dots in them. The experimental results show that the proposed method can detect C-Dots with good accuracy. In many cases, this method is successful in recognizing C-Dots with an accuracy rate above 90%. These results indicate that SVM has the potential to be an effective tool in detecting C-Dots in SEM images.

Keywords: C-Dots, Not C-Dots , SEM, SVM.

BAB I
PENDAHULUAN

I.1 Latar Belakang

Dalam beberapa kasus karbon nano-dot (C-Dots) sangat sulit untuk dapat dibedakan antara jenis carbon atau selulusa yang lainnya. Karbon nano-dot dapat didefinisikan sebagai material nano yang akan dikompositkan dengan berbagai sintetik ataupun polimer alam dengan model yang teratur. Carbon nano-dot adalah salah satu bahan karbon yang memiliki ukuran tidak lebih dari 10 nm. C-Dots diberbagai kalangan ilmuan dan universitas telah menjadi nanopartikel yang cukup banyak dikembangkan secara intensif, bahkan sampai sekarang terus dikembangkan dengan berbagai jenis penelitian baik itu sintesis C-Dots dengan sintesis yang berupa polimer alam ataupun berbagai sintesis buat untuk mengembangkan dan mengetahui apa sifat dan struktur lain dari C-Dots. Sampai sekarang penelitian tentang karbon yang menarik tentu saja masih tentang ikatan sumber karbon, namun itu tidak membatasi kita untuk memeliti nanopartikel yang satu ini lebih dalam dan lebih kreatif lagi. [1,2].

Salah satu penelitian yang menarik untuk dilakukan adalah mendeteksi karbon nanodot ini agar dapat dibedakan pada beberapa gambar, karena beberapa gambar karbon nanodot tidak dapat dibedakan antara C-dots dan selulosa tempat mereka melekat pada beberapa kasus. Untuk proses pendeteksian hal ini akan digunakan pemanfaatan citra digital. Dalam pengolahan citra digital sebenarnya sudah banyak proses atau cara yang telah dikembangkan, tujuan pengembangan metode tersebut adalah untuk bagaimana computer dapat mendeteksi dan mengenali objek pada suatu citra dengan tepat dan mengolahnya dalam waktu yang akan relative singkat. Citra sendiri dapat di artikan secara matematis sebagai fungsi kontinu dengan intensitas cahaya pada dua dimensi [3,4].

Pada gambar yang akan diolah dalam bentuk citra digital biasanya akan memiliki macam-macam warna yang berbeda artinya komposisi RGB cukup beragam. RGB secara sederhana dapat diartikan sebagai suatu model warna yang terdiri atas 3 buah warna, Red (R), Green (G), dan Blue (B). Komposisi RGB ini

yang akan mempengaruhi warna yang akan muncul pada computer untuk setiap gambar, perbedaan RGB akan dikombinasikan sehingga membentuk warna-warna lain sehingga didapatkan warna yang dimaksud pada gambar yang akan ditayangkan pada computer atau perangkat lainnya. Untuk kombinasi RGB yang sederhana dapat dijelaskan contoh untuk warna hitam nilai R=0, G=0, dan B=0 sedangkan untuk warna putih nilai RGB-nya bernilai R=255, G=255, dan B=255. Hitam dan putih adalah salah satu kombinasi warna yang mungkin dibuat oleh kombinasi RGB untuk membentuk warna yang diinginkan [5].

Pada proses citra digital akan digunakan warna citra gambar dalam bentuk grayscale. Grayscale sendiri dapat diartikan sebagai nuansa warna monokromatik dari hitam menjadi putih, sehingga grayscale diartikan sebagai warna abu-abu. Penggunaan Grayscale ini berguna untuk proses citra digital. [6].

Pada proses membedakan Carbon nano-dot dengan material lain yang mungkin tempat melekat carbon nanodot seperti selulosa dapat digunakan ccitra digital untuk membedakan bentuk-bentuk citra dari gambar yang diambil. Pada metode citra digital yang dapat dideteksi atau dibedakan hanya dalam bentuk yang sederhana seperti bagun datar pada umumnya, yaitu lingkaran, persegi Panjang, dan garis. Untuk membedakan antara lingkaran dan garis akan digunakan beberapa metode dalam citra digital dan yang cukup popular adalah Hough Transform baik untuk lingkaran maupun garis.

Pada penelitian ini akan digunakan mesin learning yaitu Support Vector mechine (SVM) metode ini akan memisahkan 2 jenis class pada sebuah data dengan mencari hyperplane terbaik pemisah 2 buah class tersebut. Sebelum menggunakan mesin learning tersebut akan digunakan metode citra digital terlebih dahulu yaitu deteksi tepi dan menggunakan hough transform untuk garis dan lingkara, karena carbon nanodot berbentuk lingakaran dan data yang digunakan dalam penelitian juga berupa selulosa dengan bentuk garis. Hasil dari semua citra digital akan dimasukan kedalam SVM untuk menjadi data sehingga dibuat hyperplane dari data tersebut. Sehingga mesin learning SVM sangat efektif dan efisien untuk membedakan antara lingakaran dan garis pada kasus membedakan carbon nanodot dengan material lainnya.

I.2 Rumusan Masalah

1. Bagaimana cara membedakan antara C-Dots dengan citra Non C-Dots dengan metode SVM?
2. Berapa besar akurasi metode SVM untuk mengklasifikasikan C-Dots dengan Citra Non C-Dots?

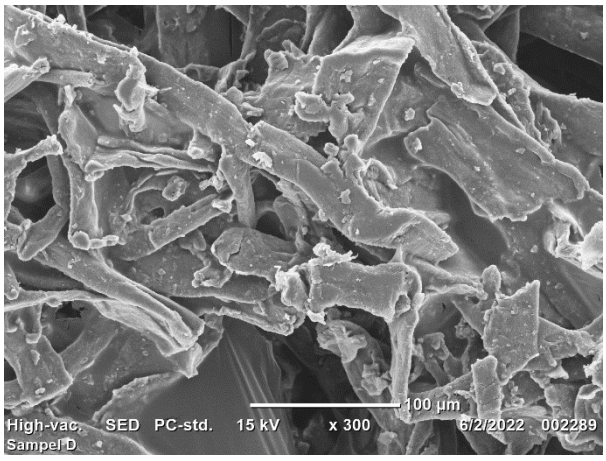
I.3 Tujuan Penelitian

1. Membuat pembelajaran mesin metode SVM untuk membedakan C-Dots dengan Citra Non C-Dots.
2. Menghitung akurasi metode SVM untuk mengkalasifikasi C-Dots dengan citra Non C-Dots.

BAB II
TINJAUAN PUSTAKA

II.1 Carbon Nanodot

Carbon nanodots (C-dots) adalah kelas material nano berbasis karbon yang relatif baru dengan ukuran di bawah 10 nm dan bentuk hampir bulat[1]. Selama bertahun-tahun, nanopartikel ini semakin populer karena sifat fotofisika dan optoelektroniknya yang menarik[2]. Selain itu, C-Dots memiliki sifat menarik lainnya seperti toksisitas rendah, biokompatibilitas, dan kelarutan yang baik hingga sangat baik dalam air dan pelarut organik polar biasa[3]. Seperti kebanyakan nanomaterial karbon, pilihan prekursor dan kondisi sintesis yang tepat sangat penting untuk menyetel sifat mereka untuk aplikasi spesifik. [4, 5]. Terutama ketika sebuah molekul digunakan sebagai prekursor, struktur kimia C-Dots yang dihasilkan dapat mencerminkan molekul precursor. Misalnya, dengan menggunakan asam amino dan molekul pembawa amina, dimungkinkan untuk memperoleh partikel nano yang kaya akan amina permukaan dan fungsi nitrogen lainnya. [8] Yang penting, gugus amino permukaan ini dapat meniru perilaku kimia katalis amina molekuler klasik, yang memungkinkan penerapan CD yang mengandung amina sebagai platform nanokatalitik dalam sintesis organik. dan akhirnya kembali ke keadaan awalnya dalam fase padat [2,9].

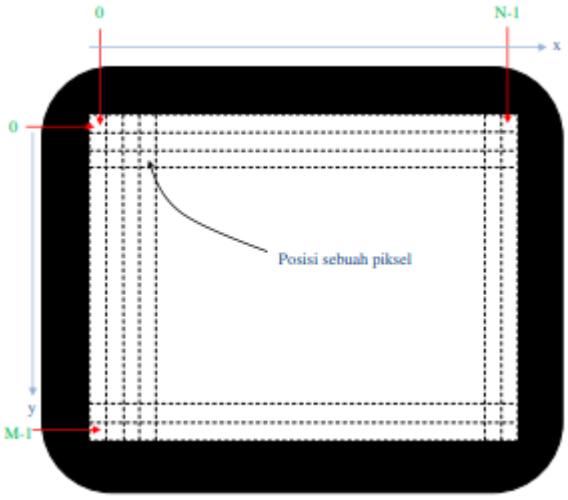


Gambar 2.1 Gambar Carbon Nanodot

II.2. Pengolahan Citra Digital

Pengolahan citra digital adalah proses manipulasi citra dengan menggunakan bantuan komputer dengan tujuan mendapatkan informasi pada gambar untuk tujuan tertentu dan proses selanjutnya.

Citra digital dibentuk dari sekumpulan piksel. Setiap piksel digambarkan sebagai kotak kecil. Citra digital didefinisikan sebagai fungsi $f(x, y)$ ukuran N kolom dan M baris, x y merupakan koordinat spasial dan f adalah titik koordinat yang menunjukkan nilai keabuan di titk tersebut[5][9][10].



Gambar 2.2 Sistem koordinat citra berukuran M x N

Berdasarkan nilai piksel, citra digital dikelompokkan kedalam tiga jenis citra yaitu:

1. Citra Warna

   Citra Warna (RGB) memiliki 3 komponen warna yakni Red, Green, Blue, di setiap piksel. Setiap komponen warna menggunakan 8 bit (nilai kisaran 0 sampai 255). Dengan demikian, kemungkinan warna yang bisa disajikan mencapai 256*256*256 atau 16.777.216 kombinasi warna.

2. Citra Aras Keabuan

   Citra aras keabuan memiliki satu nilai terusan disetiap piksel dengan kata lain nilai terusan red=green=blue. Nilai tersebut menunjukan tingkat intensitas. Tingkat intensitas dimulai dari warna hitam, keabuan sampai

putih. Citra aras keabuan memiliki kedalaman warna 8 bit (256 kombinasi warna keabuan) diperoleh menggunakan persamaan berikut :

$$aras\ keabuan = \frac{\left((red) + (green) + (blue)\right)}{3}$$

3. Citra biner, memiliki dua kemungkinan nilai piksel yakni hitam dan putih atau 0 dan 1. Citra biner sering muncul sebagai hasil proses thresholding, segmentasi ataupun morfologi.

Komponen warna citra digital setiap piksel dikonversi menjadi matriks yang berisi nilai-nilai riil. Matriks didefinisikan sebagai fungsi dua dimensi f(x,y) dengan ukuran matriks M kali N, dimana M dapat dinyatakan sebagai baris dan N adalah kolom serta x dan y adalah pasangan koordinat spasial. Nilai f pada titik koordinat (x,y) disebut sebagai skala keabuan atau intensitas dari citra digital di koordinat tersebut. Nilai x,y dan f secara keseluruhan berhingga atau bernilai diskrit sehingga citra digital. Citra digital direprentasikan dalam bentuk matriks persegi yang mewakili ukuran citra tersebut. Misalkan terdapat sebuah citra digital dengan ukuran MxN, maka citra dapat direprentasikan dalam sebuah matriks berukuran MxN sebagai berikut:

$$f(x,y) = \begin{pmatrix} f(1,1) & f(1,2) & f(1,N) \\ \dots & \dots & \dots \\ f(M,1) & f(M,2) & f(M,N) \end{pmatrix} \qquad (2.1)$$

II.3 Segmentasi Citra Otsu

Segmentasi citra bertujuan mendapatkan objek-objek citra dengan cara membagi citra ke dalam beberapa daerah yang memiliki kemiripan atribut. Terdapat dua teknik segmentasi citra yang dapat digunakan yaitu mendeteksi diskontinuitas dan similaritas. Pendekatan diskontinuitas membagi citra berdasarkan perubahan intensitas yang tiba-tiba. Sedangkan pendekatan similaritas memecah citra ke dalam daerah yang sama menurut beberapa kriteria yang sudah ditentukan, seperti proses *tersholding*.

Thresholding mengubah citra keabuan menjadi citra biner bergantung pada nilai threshold atau disebut nilai (T) sehingga dapat diketahui daerah mana yang

termasuk objek dan latar belakang. Jika piksel lebih besar dari thershould diatur menjadi 1 sebaliknya jika kurang dari threshold diatur menjadi 0. Secara umum proses thresholding ditunjukan pada persamaan 1.1[12]

$$g\,(x,y) = \begin{cases} 1\ if\ f\ (x,y) \geq T \\ 0\ if\ f\ (x,y) < T \end{cases} \tag{2.2}$$

Persamaan (2.2) menjelaskan bahwa g(x,y) adalah citra biner dari citra aras keabuan f(x,y) dan T menyatakan nilai ambang. Konsep *thresholding* Otsu pertama kali diperkenalkan oleh Nobuyuki Otsu (1979) untuk pengelompokan citra biner berdasarkan bentuk histogram secara otomatis, mengasumsikan bahwa citra berisi dua kelas dasar dengan bentuk histogram bimodal.[12] Tujuan metode Otsu adalah membagi histogram citra keabuan ke dalam dua daerah yang berbeda secara otomatis.

Berikut penurunan matematis Otsu menentukan *threshold* (k). Nilai k berkisar antara 0 sampai 255.[14]

1. Probabilitas setiap piksel pada gray level i

$$p_i = \frac{ni}{N} \tag{2.3}$$

$$n_i\ = jumlah\ piksel\ pada\ level\ ke - i$$

2. Jumlah kumulatif

$$\omega(k) = \sum_{i=0}^{k} p_i \tag{2.4}$$

3. Rerata kumulatif

$$\mu(k) = \sum_{i=0}^{k} i\ p_i \tag{2.5}$$

4. Rerata intensitas global

$$\mu T = \sum_{i=0}^{N} ip_i \tag{2.6}$$

5. Nilai ambang k ditentukan dengan memaksimumkan persamaan 2.7:

$$\sigma_B{}^2(k) = \frac{[\mu T\ \omega(k) - \mu(k)]^2}{\omega(k)[1 - \omega(k)]} \tag{2.7}$$

II.4 Deteksi Tepi Canny

Deteksi tepi dapat digunakan untuk memperoleh tepi-tepi yang diinginkan dari suatu objek citra yang digunakan. Pada batas dua daerah terdapat perubahan nilai intensitas yang cukup dratis yang disebut dengan tepi. Tepi dapat didefinisikan sebagai jarak singkat dari sebuah perubahan nilai intensitas derajat keabuan suatu citra yang mendadak (besar). Bentuk maupun ukuran objek merupakan informasi penting yang dapat diketahui dari deteksi tepi.[17] Pada deteksi tepi ada banyak teknik yang dapat digunakan, Adapun beberapa teknik dalam deteksi tepi yang dapat digunakan yaitu orde pertama menggunakan turunan pertama seperti operator *Roberts, Prewitt, Sobel* lalu orde kedua menggunakan turunan kedua seperti *Laplacian of Gaussian* (LoG) dan operator Kompas yakni medeteksi tepi kesegala arah mata angin seperti *Krish, Robinson*.[18]

Deteksi tepi Canny pertama kali dikembangkan oleh John Canny pada tahun 1986, dengan menggunakan alogaritma multi-tahap dalam mendeteksi tepi citra. Algoritma deteksi tepi Canny memiliki 3 kriteria yakni memberikan tingkat kesalahan yang paling minimum, serta mengalokasikan titik-titik tepi (jarak piksel-piksel tepi yang ditemukan deteksi dan tepi yang sesungguhnya sangat pendek), dan hanya memberikan satu respon untuk satu buah tepi. Terdapat enam tahap dalam mendeteksi tepi *Canny* sebagai berikut:[22]

1. Mengimplementasikan tapis *Gaussian* untuk mereduksi noise dan meningkatkan kualitas detail citra. Proses ini menghasilkan citra tampak sedikit lebih buram yang tujuan proses ini mendapatkan tepian citra yang sebenarnya sehingga garis-garis halus yang berada di area gradasi pada

citra masih bisa dideteksi. Filter *Gaussian* 2 dimensi dinyatakan dalam persamaan berikut:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.8)$$

Dimana G(x,y) elemen matriks gaussian pada posisi (x,y), π = 22/7, e = 2.71828182846, σ ialah nilai standar deviasi. Berdasarkan rumus (2.8), salah satu filter *Gaussian* dengan σ= 1.4 dan ukuran kernel 3x3 adalah:

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Contoh: Misalkan matriks citra berukuran 5 * 5 yang berisi:

Tabel 2.1 Contoh Matriks Deteksi Tepi Canny

0	**16**	**32**	**48**	**64**
16	**0**	**16**	**32**	**48**
32	**16**	**0**	**16**	**32**
48	**32**	**16**	**0**	**16**
64	**48**	**32**	**16**	**0**

Matriks dikalikan antar komponen dengan kernel Gaussian. Proses perkalian komponen utama di 2,2 dijelaskan sebagai berikut:

H(2,2)=(0*1+16*2+32*1+16*2+0*4+16*2+32*1+16*2+10*1)/16=12

Proses perkalian komponen utama di 4,2 dijelaskan sebagai berikut:

H(4,2)=(32*1+48*2+64*1+16*2+32*4+48*2+0*1+16*2+32*1)/16=34

Proses perkalian komponen utama di 3,4 dijelaskan sebagai berikut:

H(3,4)=(32*1+16*2+0*1+48*2+32*4+16*2+64*1+48*2+32*1)/16=34

Perkalian dikerjakan untuk semua komponen matriks citra. Hasil perhitungan adalah:

$$\begin{bmatrix}12 & 19 & 34\\ 19 & 12 & 19\\ 34 & 19 & 12\end{bmatrix}$$

2. Melakukan perhitungan besar gradient dan arah tepi. Perhitungan gradient magnitude menggunakan salah satu kernel *Roberts, Prewitt,* atau *Sobel.*
3. Memperkecil garis tepi yang muncul dengan menerapkan *nonmaximum suppression* sehingga mengahsilkan garis tepian yang terlihat lebih ramping. *Nonmaximum Suppresion* sendiri dilaksanakan dengan

mempertimbangkan dan juga memperhatikan dua titik tetangga yang terletak pada arah tepi. Jika nilai piksel titik perhatian lebih besar daripada gradien kedua tetangga, nilainya dipertahankan. Sebaliknya, jika piksel titik perhatian lebih kecil daripada nilai salah satu atau kedua gradien tetangga, nilainya dirubah menjadi 0.

4. Menerapkan dua buah *threshold* (*double thresholding*). Sederhananya *double thresholding* bertujuan klarifikasi dua buah *High-thershould* (T_2) dan *low-thershold* (T_1), dengan $T_2 \approx 2T_1$. Jika nilai piksel lebih besar atau sama dengan T_2 maka diatur nilai 255, jika nilai piksel kurang dari atau sama dengan T_1 maka diatur menjadi 0. Piksel diantara T_1 dan T_2 disebut kandidat piksel tepi maka sementara diberi nilai 128.
5. *Edge tracking by hysteresis* bertujuan memperoleh tepian final dengan menekan semua sisi yang tidak terhubung pada tepian yang sangat kuat. Nilai 128 selanjutnya dilakukan pengecekan pada piksel dari 8 arah tetangganya, sehingga piksel hanya bernilai 0 atau 255. Perubahan nilai 128 menjadi nilai 255 apabila semua kondisi terpenuhi yakni jika salah satu atau semua piksel pada 8 arah tetangganya bernilai 255. Proses pengujian dilakukan samapai tidak ada lagi perubahan dari nilai 128 menjadi 255. Selanjutnya, semua piksel yang bernilai 128 yang tersisa diubah menjadi 0.[21][22]

Tabel 2.2 Pengujian mengubah nilai 128 menjadi 255

	j-1	j	j+1				
i-1	255	255	255		255	255	255
I	255	128	255		255	255	255
i+1	255	255	255		255	255	255

II.5 Transformasi Hough

Transformasi Hough adalah sebuah transform yang pertama kali diperkenalkan oleh paul Hough pada tahun 1962, Transformasi Hough (TH) merupakan teknik pengalokasian bentuk-bentuk dalam gambar. Secara khusus, transformasi ini digunakan untuk ekstraksi garis, lingkaran dan elips. TH kemudian diimplementasikan untuk mendapatkan bangun ruang tertentu dalam

gambar dan kemudian meluas karena transformasi ini memiliki banyak kelebihan dan banyak potensi untuk pengembangan lebih lanjut. Kelebihan utamanya yaitu dapat memberikan hasil lebih cepat dan sama dengan pencocokan pola[18][4].

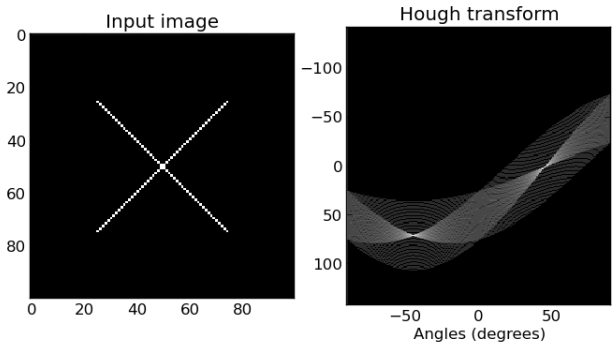
Ide awal TH melakukan pemetaan dari titik gambar menuju sebuah ruang akumulator (ruang Hough). Pemetaan tersebut diperoleh dalam bentuk yang efisien secara matematis, berdasarkan fungsi yang menjelaskan kondisi dari target. Pemetaam ini membutuhkan jauh lebih sedikit sumber perhitungan matematis daripada dengan pencocokan pola. Karena TH ekuivalen dengan pencocokan pola, TH menjadi salah satu dari teknik-teknik ekstraksi bentuk yang sering digunakan.

II.5.1 Transformasi Hough Garis

Transformasi Hough garis digunakan untuk melakukan deteksi garis lurus. Secara detail Transformasi Hough garis dapat diartikan sebagai teknik transformasi piksel dengan nilai tertentu yang dapat digunakan untuk memperoleh garis dalam sebuah citra. Ini dilakukan untuk mendapatkan suatu fitur yang lebih spesifik sesuai dengan tujuan utama dari transformasi garis ini[1][4]. Dua buah titik bisa membentuk satu garis lurus dengan persamaan:

$$y = ax + b \tag{2.9}$$

dengan nilai a dan b tertentu. Jika ada banyak titik dan dipilih hanya dua titik tertentu, terdapat banyak kombinasi pasangan dua titik yang memiliki nilai a dan b tertentu. Maka, untuk setiap pasangan a dan b tertentu, terdapat sejumlah kejadian. Garis dibentuk dari setiap pasangan a dan b yang memenuhi syarat pengambangan. Ilustrasi tersebut terdapat dalam gambar 2.3.



Gambar 2.3 Ilustrasi Transformasi Hough garis

Contoh :

Misalkan beberapa titik koordinat yang mungkin membentuk garis.

-2,2	-1,2	0,2	1,2	2,2
-2,1	-1,1	0,1	1,1	2,1
-2,0	-1,0	0,0	1,0	2,0
-2-1	-1,-1	0,-1	1,-1	2,-1
-2,-2	-1,-2	0,-2	1,-2	2,-2

Kotak yang berwarna kuning adalah garis, dan angka di dalam kotak adalah koordinat kartesius. Dalam kotak dibuat ada delapan kotak, didaftarkan semua koordinat. Lalu antar semua titiknya dihitung nilai a dan b yang mengacu pada persamaan (2.9).

Tabel 2.3 Daftar nilai a dan b setiap koordinat

No	Koordinat	Relasi	a	b
1	-2,0	P1 dan P2	-1	-2
2	-1,1	P1 dan P3	-1	-2
3	0,2	P1 dan P4	-3	-2
4	1,1	P1 dan P5	inf	inf
5	2,0	…		
6	1,-1	P2 dan P3	-1	-2
7	0,-2	P2 dan P4	inf	inf
8	-1,-1	P2 dan P5	3	2
..	…..	…	….	….
28	-1,-1	P7 dan P8	-1	2

Lalu membuat koordinat a dan b semua kejadian pasangan nilai a dan b tertentu dituliskan dalam koordinat tersebut. Garis akan dibentuk pada kejadian yang paling besar.

		b								
		-4	-3	-2	-1	0	1	2	3	4
a	-4	0	0	0	0	0	0	0	0	0
	-3	0	0	0	0	0	0	0	0	0
	-2	0	1	0	0	0	0	0	1	0
	-1	0	0	2	0	0	0	2	0	0
	0	0	0	0	1	1	1	0	0	0
	1	0	0	2	0	0	0	2	0	0
	2	0	1	0	0	0	0	0	1	0
	3	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0

Jumlah kejadian terbanyak bernilai 2 ada 4

a	b
-1	-2
1	-2
-1	2
1	2

Gambar 2.4 Kontur Kejadian Garis

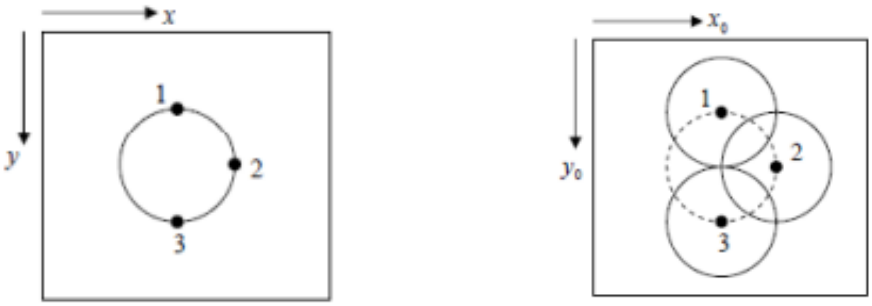
Sehingga didapatkan kejadian terbesar ada pada koordinat (-1,-2), (1,-2), (-1,2), dan (1,2) yang masing-masing sebanyak 2 kejadian.

II.5.2 Transformasi Hough Lingkaran

Transformasi Hough lingkaran berawal dari persamaan umum lingkaran, berikut:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \qquad (2.10)$$

Di perasamaan (2.10) ada tiga tetapan yaitu xo, yo dan r, ada dua variable yaitu x dan y yang menunjukkan koordinat piksel lingkaran yang terdapat. Satu pasang koordinat x dan y dari gambar dan nilai r yang dipilih dipetakan ke titik pusat lingkaran pada sudut tertentu. Hasil pemetaan ini adalah titik pusat lingkaran dan jari-jari yang ditetapkan. Untuk banyak titik piksel dan jari-jari yang telah dipilih terdapat banyak hasil pemetaan. Titik pusat lingkaran diambil dari kejadian yang memenuhi ambang batas tertentu.



(a) (b)

Gambar 2.5 (a) Citra yang berisi lingkaran (b) Ruang Akumulator

Contoh:

R=1, (a,b)=(0,0)		
Pilihan r = 1		
		(a,b)
4 titik yang dilewati =		(-1,0)
		('0,1)
		(-1,0)
		(0,-1)

Tabel 2.4 Daftar nilai a dan b untuk setiap scan sudut

Scan tiap 90	dipilih	r = 1				
0, 90, 180, 270,360						
		θ				
		0	90	180	270	360
a,b	-1,0	0,0	1,1	-2,0	-1,-1	0,0
	0,1	1,1	0,2	-1,1	0,0	1,1
	1,0	2,0	1,1	0,0	1,-1	2,0
	0,-1	1,-1	0,0	-1,-1	0,-2	-1,-1

Misalkan lingkaran dengan jari-jari 1 dan titik pusat (0,0). Ada 4 titik yang mungkin dilewati lingkaran. Lalu kita pilih jari-jari = 1 scan sudut 0, 90, 180, 270, 360 dan titik pusatnya (-1,0), (0,1), (1,0), (0,-1). Misalkan titik pusat (-1,0) pada sudut 0 derajat memiliki titik lingkaran (0,0) dan seterusnya sesuai dengan scan. Akan didapatkan gambar untuk setiap kejadian sebagai berikut:

		b				
		-2	-1	0	1	2
a	2	0	0	0	0	0
	1	0	**2**	**1**	**2**	0
	0	**1**	0	**4**	0	**1**
	-1	0	**2**	**1**	**2**	0
	-2	0	0	0	0	0

Gambar 2.6 Daftar kejadian terbesar

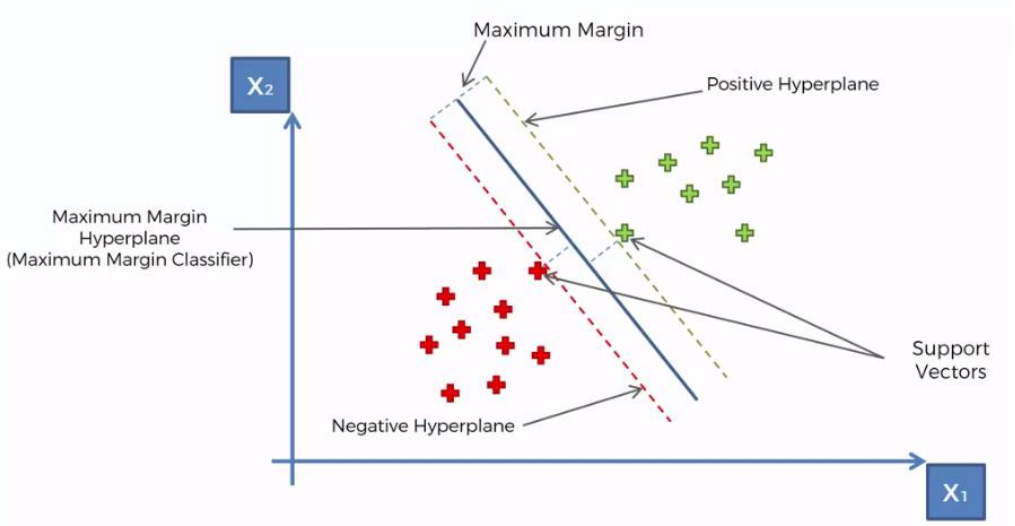
Gambar 2.5 menyatakan koordinat dengan kejadian terbanyak yaitu (0,0) sebanyak 4 kejadian, artinya koordinat (0,0) menjadi pusat lingkaran.

II.6 Support Vector Machine (SVM)

Support vector machine (SVM) adalah model mesin pembelajaran dengan algoaritma yang mempelajari menganilisis data untuk klasifikasi dan analisis regresi. SVM dikembangkan AT&T Bell Laboratories oleh Vladimir Vapnik dengan temannya. SVM adalah salah satu metode prediksi paling kuat, didasarkan pada kerangka pembelajaran statistik atau teori VC yang diusulkan oleh Vapnik (1982) dan Chervonenkis (1974). Diberikan satu set contoh pelatihan, masing-masing ditandai sebagai milik salah satu dari dua kategori, algoritma pelatihan SVM membangun model yang memberikan contoh baru ke satu kategori atau yang lain, menjadikannya sebagai pengklasifikasi linier biner

non-probabilistik (walaupun metode seperti Platt penskalaan ada untuk menggunakan SVM dalam pengaturan klasifikasi probabilistik). SVM memetakan contoh-contoh pelatihan ke titik-titik dalam ruang untuk memaksimalkan lebar celah antara kedua kategori tersebut. Contoh-contoh baru kemudian dipetakan ke dalam ruang yang sama dan diprediksi termasuk dalam kategori berdasarkan di sisi celah mana mereka jatuh.

SVM berusaha untuk mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada *input* yang diinginkan. *Hyperplane* adalah sebuah fungsi yang dapat digunakan untuk pemisah antar kelas. Pemisahan yang dibuat oleh OSH (*Optimum Separating Hyperplane*), menjadi pemisah yang sangat optimal yang dapat digunakan untuk klasifikasi. Misalkan $\{x_1, \ldots\ldots, x_n\}$ adalah kumpulan data dan $y_i \epsilon \{+1, -1\}$ adalah sebuah kelas dari data[11][21].



Gambar 2.7 SVM dan OSH

Pada gambar 2.6 dapat dilihat bidang pemisah yang memisahkan semua objek sesuai dengan kelasnya. Data yang di perbatasan area disebut Support Vector. Dua kelas dapat dipisahkan oleh sepasang bidang pembatas yang sejajar. Pertama terlihat *hyperplane* membatasi kelas pertama sedangkan *hyperplane* kedua membatasi kelas kedua, sehingga dapat dibuat persamaan: [11][21]

$$x_i . w + b \geq +1, y_i = +1 \quad (2.11)$$

$$x_i . w + b \geq +1, y_i = -1 \quad (2.12)$$

w adalah Normal plane atau yang disebut sebagai *support vector* dan b adalah posisi relative bidang terhadapt pusat koordinat atau yang disebut juga dengan bias. Karena data berupa nonlinear, maka data akan diubah menjadi fitur dimensi menggunakan fungsi pemetaan (transformasi) $x_k \rightarrow \emptyset(x_k)$ dengan a sebagai fungsi kernel K, kernel yang digunakan pada persamaan ini adalah kernel RBF yang dapat dilihat pada persamaan berikut:[21]

$$k(x_i, x_j) = \exp\left(-\frac{1}{s^2}\left\|x_i - x_j\right\|^2\right) \tag{2.13}$$

Dimana x_i dan x_j adalah *feature vectors* dan δ adalah produk dari parameter C dan gamma SVM dapat diatur oleh user. Persamaan Weight (W) vector dapat ditulikan dalam definiskan berikut:[11]

$$W = \sum_{i=1}^{N} \alpha_i y_i k(x_i) \tag{2.14}$$

Weight vector biasanya akan memiliki nilai yang sangat besar, namun hal ini bergantung pada nilai α. Lagrange multiplier digunakan untuk menetukan nilai dari α, sehingga kita dapat menentukan nilai w. Nilai b dapat dditentukan dari persamaan berikut:[21]

$$b = \frac{1}{SV} \sum_{x_i \in SV} \left(\frac{1}{y_i} - \sum_{x_j \in SV} \alpha_j y_j k(x_{j,} x_i) \right) \tag{2.15}$$

Dimana SV adalah jumlah Support vector. Separator terbaik pada persamaan hyperplane adalah program persamaan kuadrat dengan nilai maximum sehingga kita bisa mencari nilai α. Setelah persamaan kuadrat yang diingikan ditemukan maka kelas dari berbagai nilai x dapat ditentukan persamaan determinan fungsi sebagai berikut: [15][21]

$$D(z) = sign\left(\sum_{i=1}^{N} \alpha_i y_i k(x_{i,} z) + b\right) \tag{2.16}$$

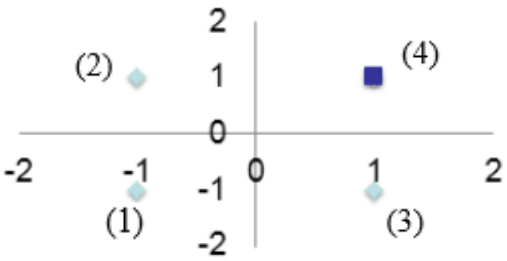
Dimana D(z) adalah persamaan *decision* untuk semua kelas dan N adalah angka untuk support vector.[15]

Misalkan ada data seperti pada tabel 2.5 untuk setiap koordinat yang dicari *hyperplane*:

Tabel 2.5 Tabel Contoh Parameter

x	Y	N
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

Keempat data di dalam tabel 2.5 dapat diilustrasikan dalam gambar (2.7)



Gambar 2.8 Ilustrasi tabel 2.5 dalam koordinat kartesian

Keterangan gambar:

$$w_1.x + w_2.y + b \geq 1, termasuk\ kelompok\ persegi$$

$$w_1.x + w_2.y + b \leq -1, termasuk\ kelompok\ belah\ ketupat$$

Dapat digunakan persamaan:

$$y_i.(x_i.w + b) \geq 1 \quad (2.17)$$

$$N_i(x_i\, w_x + y_i w_y + b) = 1 \quad (2.18)$$

Nilai i di persamaan 2.18 berarti baris matriks, dan nilai x, y, N dimasukan ke persamaan 2.18. Dengan memasukkan nilai dalam tabel ke persamaan (2.17), didapatkan empat persamaan:

$$-w_x - w_y + b = -1$$

$$-w_x + w_y + b = -1$$

$$w_x - w_y + b = -1$$

$$w_x + w_y + b = 1$$

Solusi persamaan 2.18 dapat dilakukan dengan metode subtitusi. Untuk nilai x=-1, y=-1 dan N=-1, didapatkan:

$$(-1)((-1)w_x + (-1)w_y + b) \geq 1$$

$$w_x + w_y - b \geq 1$$

Untuk nilai x=-1, y=1 dan N=-1, didapatkan:

$$(-1)((-1)w_x + (1)w_y + b) \geq 1$$

$$w_x - w_y - b \geq 1$$

Untuk nilai x=1, y=-1 dan N=-1, didapatkan:

$$(-1)((1)w_x + (-1)w_y + b) \geq 1$$

$$-w_x + w_y - b \geq 1$$

Untuk nilai x=1, y=1 dan N=1, didapatkan:

$$(1)((1)w_x + (1)w_y + b) \geq 1$$

$$w_x + w_y + b \geq 1$$

Kemudian dilakukan eliminasi:

$$w_x + w_y - b = 1$$

$$w_x - w_y - b = 1$$

---------- **(-)**

$$2w_y = 2$$

$$\boldsymbol{w_y = 1}$$

$$w_x + w_y + b = 1$$

$$w_x - w_y - b = 1$$

---------- **(+)**

$$2w_x = 2$$

$$\boldsymbol{w_x = 1}$$

$$-w_x - w_y - b = 1$$

$$w_x - w_y - b = 1$$

---------- **(+)**

$$-2b = 2$$

$$\boldsymbol{b = -1}$$

Maka didapatkan persamaan *hyperplane* sebagai berikut:

$$w_x.x + w_y.y + b = 0$$

$$1x + 1y + (-1) = 0$$

$$\boldsymbol{x + y = 1}$$

Selain itu, solusi menggunakan aturan Cramer dihitung menggunakan 2 kombinasi pengambilan titik.

Kombinasi pertama diambil titik (2), (3), (4)

Dihitung determinan yang menghasilkan $\Delta = 4, \Delta Wx = 4, \Delta Wy = 4, \Delta b = -4$.

Maka nilai $w_1 = 1$; $w_2 = 1$; $b = -1$

Persamaaan hyperplanenya adalah

$$\boldsymbol{x + y = 1}$$

Untuk titik 1 (-1, -1)

$$1\,(-1) + 1(-1) - 1 = -3$$

Karena N di titik $1 \leq -1$, titik satu merupakan kelompok belah ketupat.

Untuk titik 2 (-1,1)

$$1(-1) + 1(1) - 1 = -1$$

Karena N di titik $2 \leq -1$, titik dua merupakan kelompok belah ketupat.

Untuk titik 3 (1, -1)

$$1(1) + 1(-1) - 1 = -1$$

Karena N di titik $3 \leq -1$, titik tiga merupakan kelompok belah ketupat.

Untuk titik 4 (1,1)

$$1(1) + 1(1) - 1 = 1$$

Karena N di titik $4 \geq 1$, titik empat merupakan kelompok persegi.

Kombinasi kedua diambil titik (1), (2), (4)

Dihitung determinan yang menghasilkan $\Delta = -4, \Delta Wx = -4, \Delta Wy = 0, \Delta b = 0$.

Maka nilai $w_1 = 1$; $w_2 = 0$; $b = 0$

Persamaaan hyperplanenya adalah

$$\boldsymbol{x = 0}$$

Untuk titik 1 (-1, -1)

$$1\,(-1)\ +\ 1(0)\ -0\ =\ -1$$

Karena N di titik $1 \leq -1$, titik satu merupakan kelompok belah ketupat.

Untuk titik 2 (-1,1)

$$1(-1) + 1(0) - 0 = -1$$

Karena N di titik $2 \leq -1$, titik dua merupakan kelompok belah ketupat.

Untuk titik 3 (1, -1)

$$1(1) + 1(0)\ -0\ =\ 1$$

Karena N di titik $3 \geq 1$, titik tiga merupakan kelompok persegi.

Untuk titik 4 (1,1)

$$1(1) + 1(0)\ -\ 0\ =\ 1$$

Karena N di titik $4 \geq 1$, titik empat merupakan kelompok persegi.

II.7 *Confusion Matriks*

Confusion matriks adalah salah satu metode yang sering digunakan dalam melakukan perhitungan akurasi pada konsep *data mining*. *Confusion matrix* digambarkan dengan menyatakn jumlah data uji yang benar diklasifikasikan dalam jumlah data uji yang salah diklasifikasikan. Evaluasi dengan confusion matrix menghasilkan nilai *accuracy, precision, dan recall*.[23].

Tabel 2.6 *Confusion Matrix*

Correct Classification	*Classified as*	
	Predicted "+"	*Predicted "-"*
Actual "+"	*True Positive*	*False Negative*
Actual "-"	*False Positive*	*True Negative*

Beradasarkan tabel 2.6 *Confusion matrix* diatas, *true positive* (TP) adalah jumlah record data positif yang diklasifikasikan sebagai positif, *False Positive* (FP) adalah jumlah record data negative yang diklasifikasikan sebagai nilai positif, *False Negative* (FN) adalah jumlah record data positif yang diklasifikasikan, sebagai nilai positif, dan *true negative* (TN) adalah jumlah record data negative yang diklasifikasikan sebagai nilai negatif. [23]

Membuat tabel perbandingan actual dan prediksi.

x	Y	N(Actual)	N(Cramer 1)	N(Cramer 2)
-1	-1	-1	-3	-1
-1	1	-1	-1	-1
1	-1	-1	-1	1
1	1	1	1	1

Dari kombinasi pertama metode Cramer dapat dibuat *Confusion matrix*

		Prediksi	
		◆	▭
Aktual	◆	3	-
	▭	-	1

Dari kombinasi kedua metode Cramer dapat dibuat *Confusion matrix*

		Prediksi	
		◆	▭
Aktual	◆	2	1
	▭	-	1

Dengan aturan Cramer, kombinasi titik yang diambil mempengaruhi *Conffusion matriks.*

BAB III
METODE PENELITIAN

III.1 Python

Python adalah salah satu bahasa pemerograman tingkat tinggi yang bersifat interpreter, interaktif, berorientasi objek dan dapat beroperasi di *platform* Linux, Windows, Mac dan *platform* lainnya. Python adalah salah satu bahasa pemerograman tingkat tinggi yang mudah dipelajari karena sintaks yang jelas dan elegan, yang dikombinasikan dengan penggunaan modul-modul yang mempunyai struktur data tingkat tinggi, efisien dan siap langsung digunakan. *Source code* aplikasi dalam bahasa pemrograman Python biasanya dikompilasi menjadi format perantara yang dikenal sebagai *bytecode* yang selanjutnya diekseskusi. Karena keunggulan-keunggulan Python di atas, Python dipilih menjadi bahasa pemrograman yang akan digunakan dalam proses deteksi citra digital untuk deteksi C-Dot pada kasus ini.

III.2 Alat dan bahan

Alat yang digunakan dalam penelitian ini antara lain:

1. Satu set portable computer (PC) dengan operating system (OS) Windows 10 atau lebih.
2. Aplikasi editor bahasa pemograman Python.

Bahan yang digunakan dalam penelitian ini antara lain:

1. Citra SEM Karbon nano-dot (C-Dots)
2. Citra SEM material lain.

III.3 Konversi Citra

Lima citra SEM karbon nano-dot dan lima citra SEM material lain, diunduh dari website untuk dijadikan input. Piksel citra tersebut diubah ukurannya menjadi 200 x 200 piksel. Modul cv2 diinstalasi masuk kedalam Pyhton. Fungsi

COLOR_RGB2GRAY digunakan untuk mengubah citra ke matriks derajat keabuan.

III.4 Deteksi Tepi Canny

Deteksi tepi Canny dikenakan pada citra. Sebagai input adalah matriks derajat keabuan. Supaya Python dapat melakukan deteksi tepi Canny, modul cv2 dimasukan ke dalam Python. Untuk mengeksekusi deteksi tepi Canny, matriks citra keabuan dimasukkan fungsi Canny. Mekanisme deteksi tepi Canny dijelaskan dalam subbab (II.4). Keluaran deteksi tepi Canny adalah martiks yang berisi angka 0 dan 255. Piksel bernilai 255 berarti piksel tepi dan piksel bernilai 0 berarti bukan piksel tepi.

III.5 Transformasi Hough Garis

Pada penelitian ini digunakan transformasi Hough garis untuk mendeteksi objek-objek garis. Supaya Python dapat melakukan transformasi Hough garis, modul cv2 dimasukan ke dalam Python. Matriks citra hasil deteksi tepi Canny dimasukkan ke dalam fungsi HoughLines di Python. Mekanisme transformasi dijelaskan dalam subbab (II.5.1) Keluaran transformasi dari fungsi HoughLines adalah jumlah garis dan sudut garis.

III.6 Transformasi Hough Lingkaran

Pada penelitian ini digunakan transformasi Hough lingkaran untuk mendeteksi objek-objek lingkaran. Supaya Python dapat melakukan transformasi Hough lingkaran, modul cv2 dimasukan ke dalam Python. Matriks citra hasil deteksi tepi Canny dimasukkan ke dalam fungsi HoughCircles di Python. Mekanisme transformasi dijelaskan dalam subbab (II.5.2) Keluaran transformasi dari fungsi HoughCircles adalah jumlah lingkaran dan jari-jari setiap lingakaran.

III.7 Metode SVM

Input yang digunakan untuk menentukan jenis citra adalah nilai ambang batas Otsu, jumlah piksel hasil segmentasi Otsu, jumlah lingkaran, rata-rata jari-jari lingkaran, jumlah garis, dan rata-rata panjang garis. Keenam input itu dimasukkan ke dalam metode SVM yang rumusnya terdapat dalam subbab (II.6) Modul sklearn dan matplotlib dimasukan ke dalam program. Modul sklearn untuk

menghitung SVM dan modul matplotlib digunakan untuk membuat grafik. Keluaran metode ini adalah citra nano-dot atau bukan citra nano-dot.

III.8 Uji Validitas

Nilai yang dihasilkan melalui metode *confusion matrix* adalah berupa evaluasi sebagai berikut:

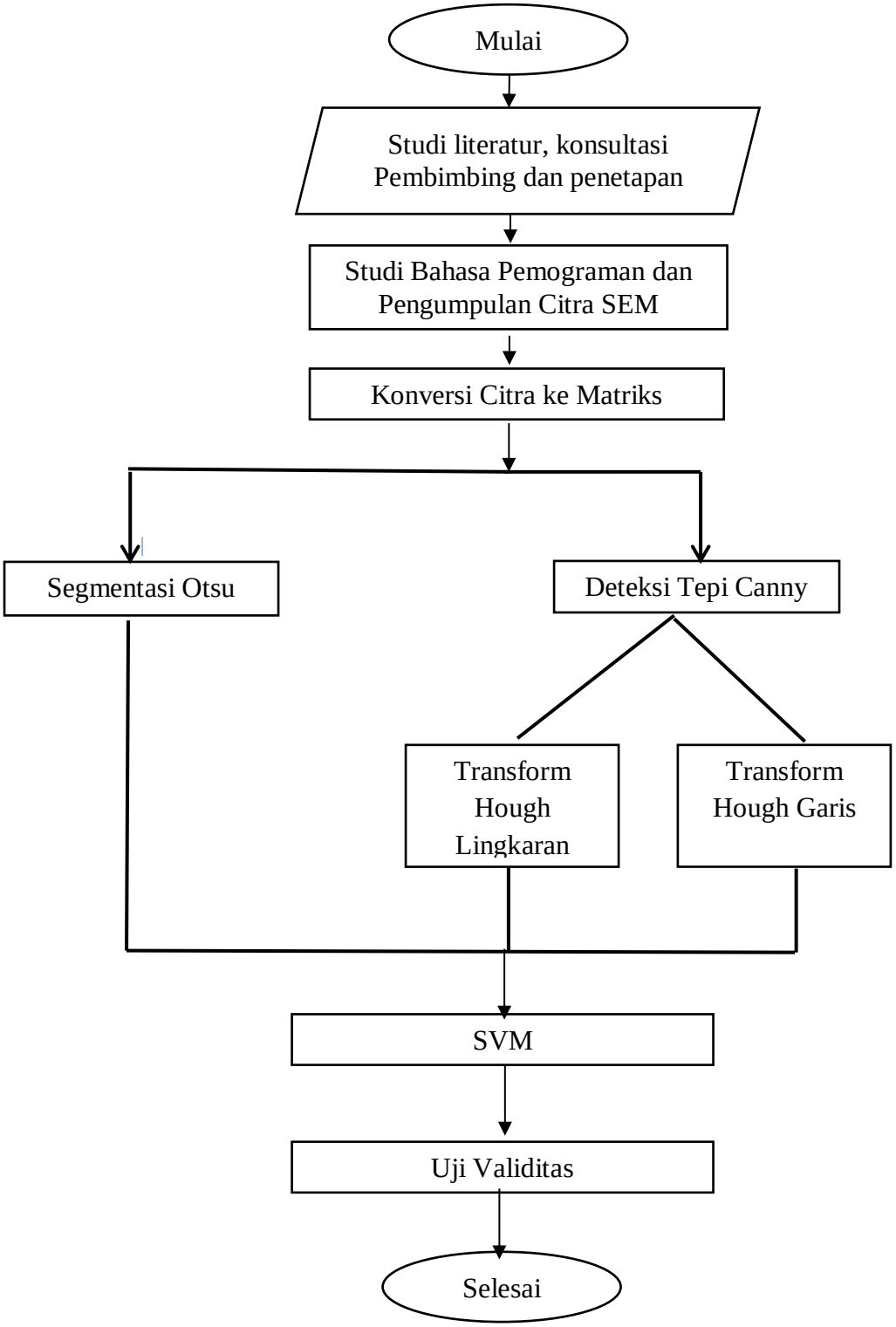
1) *Accuracy* adalah presentasi jumlah record data yang diklasifikasikan (prediksi) secara benar oleh algoritma.

$$ACC = \frac{(TP + TN)}{Total\ Data} \tag{3.1}$$

2) Misclassification Rate adalah presentase jumlah record data yang diklasifikasikan (prediksi) secara salah oleh algoritma.

$$ERR = \frac{(FP + FN)}{Total\ Data} \tag{3.2}$$

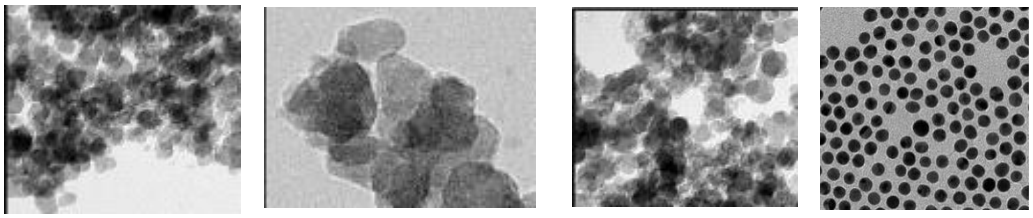
III.9 Bagan Alir Penelitian



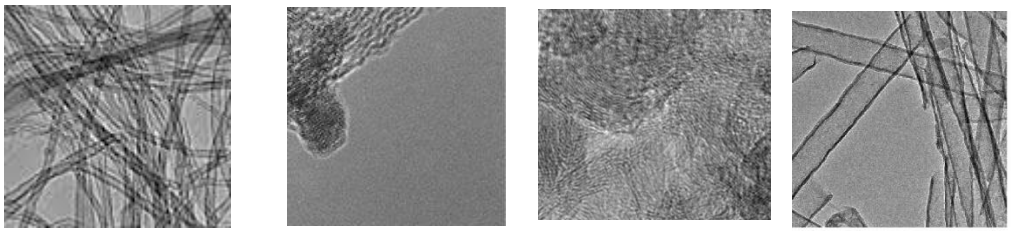
BAB IV
HASIL DAN PEMBAHASAN

IV.1 Gambar C-Dots dan Non-C-Dots

Pada penelitian ini digunakan gambar TEM C-Dots yang menjadi gambar pada penelitian ini, selanjutnya dibandingkan dengan gambar TEM Non C-Dots. Digunakan 80 sampel gambar TEM C-Dots dan 80 sampel gambar Non C-Dots. Beberapa contoh sampelnya dapat dilihat pada gambar 4.1 dan gambar 4.2.



Gambar 4.1 Karbon Nano Dot



Gambar 4.2 Non C-Dots

IV.2 Modul Python

IV.2.1 Library Python

Pada penelitian ini, digunakan berbagai modul python untuk mendapatkan hasil yang diinginkan. Library Python yang digunakan untuk membaca gambar pada Python adalah open CV. Library python cv2.imread(imageName,0) berfungsi untuk membaca matriks piksel angka pada gambar dan gambar harus berada dalam direktori kerja yang sama. Image merupakan keluaran fungsi, imageName merupakan nama gambar, dan 0 merupakan sytanx memilih keluarannya sebagai *grayscale*.

Sementara itu untuk library Python yang berfungsi untuk metode otsu adalahsintaks:ret,th=cv2.thershold(image,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU). Ret merupakan akan menampung nilai threshold yang dihitung

oleh fungsi “cv2.threshold()”, th merupakan menampung gambar hasil thresholding yang sudah diaplikasikan, Image merupakan parameter pertama dari fungsi “cv2.threshold()”, yaitu gambar input yang akan diberikan thresholding., 0 merupakan yaitu nilai thresholding yang digunakan untuk memisahkan piksel menjadi nilai yang lebih rendah atau lebih tinggi dari nilai thresholding. Nilai 0 digunakan dalam contoh ini karena tidak akan mempengaruhi hasil thresholding, 255 merupakan nilai maksimum yang digunakan untuk menetapkan nilai maksimum piksel yang lebih besar dari nilai thresholding, cv2.THRESH_BINARY merupakan akan menetapkan nilai maksimum pada piksel yang lebih besar dari nilai thresholding, cv2.THRESH_OTSU merupakan akan memilih nilai thresholding secara otomatis dengan menggunakan metode Otsu. Pada Otsu secara khusus akan digunakan modul cv2.threshold, dimana modul ini mengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk objek dan background dari citra secara jelas.

Matplotlib adalah *library* python yang berfokus pada visualisasi data seperti plot grafik baik dalam dua dimensi atau tiga dimensi. Visualisasi dalam matplotlib adalah sebuah grafik yang terdapat pada satu atau lebih sumbu. Setiap sumbu memiliki horizontal (x) dan sumbu vertikal (y), yang direpresentasikan menjadi warna dan glyphs seperti line, polygon, atau histogram yang akan mempermudah pembacaan hasil. Ploting visualiasi ini secara spesifik digunakan modul matplotlib.pyplot yang selanjutnya disingkat plt.

Sementara itu modul numpy digunakan untuk melalukan fungsi copy atau salin, numpy juga digunakan dalam proses *scientific computing* yaitu melakukan perhitungan pada matriks dan aljabar. Untuk modul numpy yang selanjutnya disingkat np akan berfungsi secara khusus untuk menyalin array dari gambar, sekaligus melakukan perhitungan pada matriks hasil citra digital yang didapatkan dari gambar.

Pada transformasi Hough pada garis digunakan modul cv2.HoughLines, dimana modul ini berfungsi untuk mendeteksi banyaknya jumlah garis, panjang

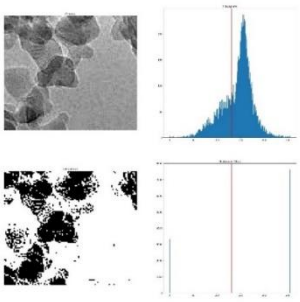
garis, nilai a dan b, dan titik awal dan titik akhir yang ada pada gambar citra yang selanjutnya diolah lebih lanjut.

Pada transformasi Hough pada lingkaran digunakan modul cv2.HoughCircles, dimana modul ini berfungsi untuk mendeteksi banyaknya jumlah lingkaran, jari-jari, dan titik pusat yang ada pada gambar citra yang selanjutnya diolah lebih lanjut.

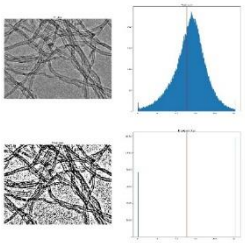
Selanjutnya pada SVM digunakan modul sklearn untuk menghitung SVM dan modul matplotlib digunakan untuk membuat grafik. Selain itu pada SVM digunakan modul Seaborn yang berfungsi sebagai salah satu library pada Python untuk visualisasi data.

IV.3 Hasil Thresholding (Otsu)

Setelah dilakukan proses *thresholding* pada semua gambar C-Dots dan Non C-Dots, didapatkan semua hasil gambar *thresholding* untuk semua gambar C-Dots dan Non C-Dots. Ssebagai contoh diberikan beberapa gambar seperti pada gambar 4.3 dan 4.4.



Gambar 4.3 Citra *Thresholding* C-Dots

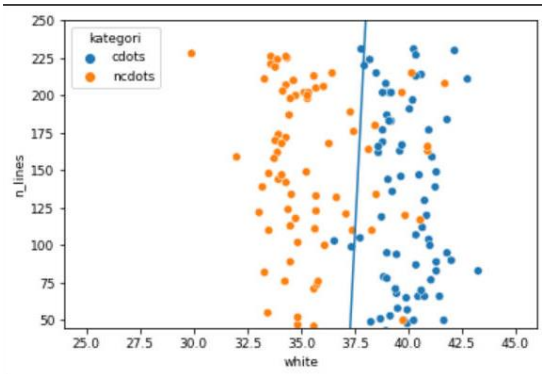


Gambar 4.4 Citra *Thresholding* Non C-Dots

Keluaran fungsi cv2.HoughCircles Adalah 4 gambar. Gambar kiri atas gambar asli, gambar kanan atas adalah distribusi histogram citra, gambar kiri bawah adalah gambar biner threshold Otsu, dan gambar kanan bawah adalah garis batas Otsu dan jumlah piksel nol dan 255. Data selengkapnya terdapat pada lampiran C.

IV.4 SVM 2 Parameter

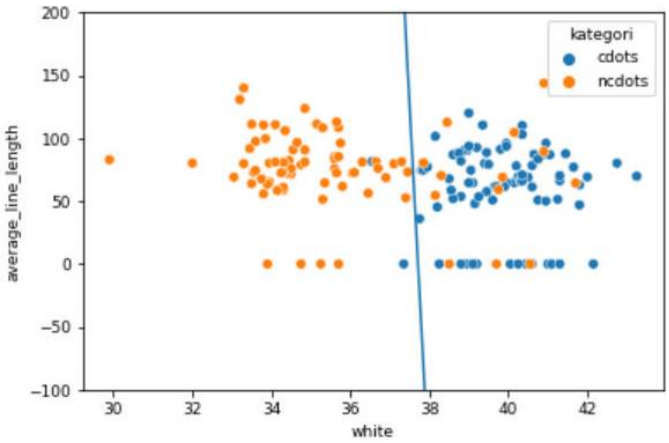
SVM digunakan modul sklearn untuk menghitung SVM dan modul matplotlib. Atribut SVM yaitu black,white, jumlah garis, rata-rata pajang garis, jumlah lingkaran, rata-rata jari-jari lingkaran, dan batas otsu. Atribut itu diplot pada SVM sehingga menghasilkan hubungan satu sama lain sehingga dapat dilihat perbandingan dari setiap 2 parameter pembanding dari setiap plot yang diberikan. Hasil plotting yang diberikan akaan memperlihatkan pemisahan untuk setiap data yang merupakan C-dots dan data yang erupakan Non C-dots, dimana pada grafik untuk yang berwarna Biru adalah data c-dots dan yang berwarna *orange* adalah data non c-dots. Contohnya adalah perbandingan jumlah garis dengan white yang grafiknya terdapat pada gambar 4.5



Gambar 4.5 Sebaran Data Jumlah Garis dan White

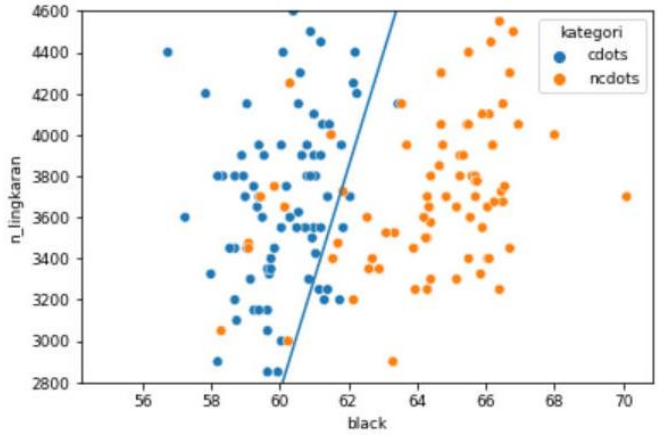
Pada gambar 4.5 terlihat bahwa cdots dan non c-dots terpisah dan tampak garis *hyperplane*. Pada kasus ini didapatkan akurasi *coffusion matrix* pada angka 90%.

Selanjutnya dapat ditinjau pada parameter lain yaitu parameter yang mungkin menjadi pemisah terbaik pada SVM.



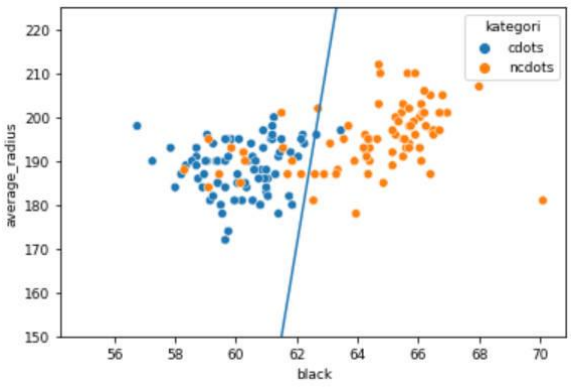
Gambar 4.6 Sebaran data rata-rata Panjang garis dan White

Pada gambar 4.6 memperlihatakan untuk perbandingan 2 parameter yaitu rata-rata jumlah garis dan white. Dapat terlihat bahwa yang menjadi pemisah terbaik pada grafik ini masih ada pada white seperti penjelasan pada perbadingan parameter jumlah garis dan white. Persamaan hyperplane dihitung secara manual yang menghasilkan akurasi *conffusion matrix* pada angka 90%.



Gambar 4.7 Sebaran data untuk Parameter black dan jumlah garis

Sedangkan untuk 2 parameter yaitu black dan jumlah garis, ternyata pemisah terbaik adalah black. Terlihat untuk c-dots nilai black ada diantara 50 sampai 60, sedangkan untuk non c-dots nilainya ada diantara 60 sampai 70. Persamaan *hyperplane* dihitung secara manual yang menghasilkan akurasi *confusion matrix* pada angka 90%.



Gambar 4.8 Rata-rata jari jari lingkaran dan black

Ketiga grafik diatas menunjukan perbedaan pada pamameter pada sumbu y, namun untuk sumb x nya sama yaitu black. Sekaligus pada keempat grafik menunjukan bahwa pemisah terbaik dari setiap grafik adalah black dengan nilai yang sama untuk gambar 4.7. Persamaan hyperplane dihitung secara manual yang menghasilkan akurasi *confusion matrix* pada angka 90%.

IV.5 SVM Tujuh Parameter

SVM tujuh parameter membutuhkan tujuh data supaya dapat diolah, supaya hasil imbang dipilih empat gambar c-dots dan tiga gambar non c-dots atau sebaliknya. Ada empat puluh citra c-dots diambil empat citra, maka ada C_4^{40}kombinasi atau yang bernilai 91.390 kombinasi. Ada empat puluh citra non c-dots diambil tiga citra, maka ada C_3^{40}kombinasi atau yang bernilai 9.880 kombinasi. Total kombinasi untuk mengolah semua data adalah 902.933.200. Di skripsi ini hanya dikerjakan seratus kombinasi saja. Maka hasil-hasil tidak dapat dianalisis karena kombinasi yang dikerjakan terlalu kecil dari kombinasi total. Pada tujuh parameter didapatkan nilai b sebesar -23,71534146613781 dan didapatkan juga tujuh nilai W yang dapat dilihat pada tabel berikut.

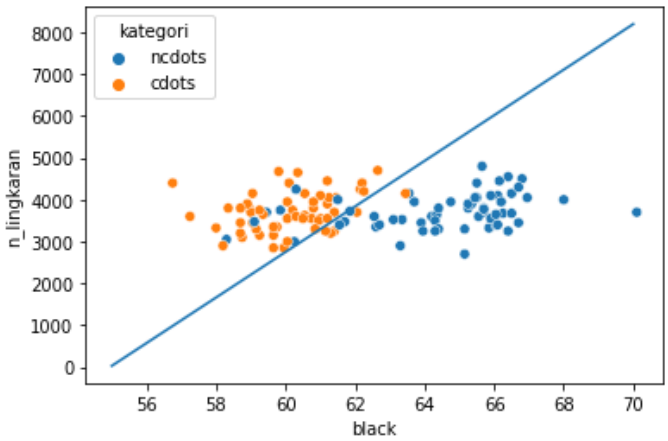
Tabel 4.1 Besar Nilai W

W_1	0,01074186
W_2	-0,02822032
W_3	0,88803298
W_4	-0,88803298

W_5	0,0872759
W_6	-0,00294188
W_7	- 0,02721059

IV.6 Hasil SVM

Setelah data dari ke-7 parameter dianalisis untuk mendapatkan 2 parameter yang mungkin baik dalam penentuan hyperplane terbaik dari data yang dimiliki, didapatkan bahwa parameter baik dalam memisahkan persebaran c-dots dan non-cdots adalah jumlah lingkaran dan black. Kedua parameter tersebut dengan sangat baik dapat memisahkan 2 data tersebut dengan akurasi 97%.



Gambar 4.9 Hasil SVM

BAB V
KESIMPULAN DAN SARAN

V.1 Kesimpulan

Dari hasil dan pembahasan yang diperoleh dapat disimpulkan bahwa;

1. Pada metode SVM cara terbaik dalam membedakan c-dots dan non c-dots adalah dengan perbandingan 2 parameter yaitu jumlah lingkaran dan black, dengan 2 parameter ini didapatkan hyperplane terbaik dari sebuah parameter sehingga dapat terlihat persebaran c-dots dan non c-dots berbeda.
2. Setelah dilakukan uji akurasi dengan metode *confusion matrix* didapatkan akurasi metode SVM ini dalam mengkasifikasikan antara c-dots dan non c-dots adalah 97%. Sehingga hasil ini membuktikan bahwa metode SVM ini mampu membedakan c-dots dan non c-dots.

V.2 Saran

Penelitian selanjutnya sebaiknya menggunakan *dataset* dari satu sumber sehingga mempercepat proses penelitian dan menggunakan teknik lainnya.

DAFTAR PUSTAKA

[1] Y. F. Fung, H. Lee, and M. F. Ercan, “Image processing application in toll collection,” Lect. Notes Eng. Comput. Sci., no. April 2015, pp. 584–588, 2006.

[2] M. Hutter and N. Brewer, “Matching 2-D ellipses to 3-D circles with application to vehicle pose identification,” 2009 24th Int. Conf. Image Vis. Comput. New Zealand, IVCNZ 2009 - Conf. Proc., pp. 153–158, 2009, doi: 10.1109/IVCNZ.2009.5378421.

[3] A. Bhujbal and D. Mane, “A survey on deep learning approaches for vehicle and number plate detection,” Int. J. Sci. Technol. Res., vol. 8, no. 12, pp. 1378–1383, 2019.

[4] A. O. Djekoune, K. Messaoudi, and K. Amara, “Incremental circle hough transform: An improved method for circle detection,” Optik (Stuttg)., vol. 133, pp. 17–31, 2017, doi: 10.1016/j.ijleo.2016.12.064.

[5] V. A. Gunawan and L. S. A. Putra, “Comparison of American Sign Language Use Identification using Multi-Class SVM Classification, Backpropagation Neural Network, K - Nearest Neighbor and Naive Bayes,” Teknik, vol. 42, no. 2, pp. 137–148, 2021, doi: 10.14710/teknik.v42i2.36929.

[6] J. Jumadi and D. Sartika, “Pengolahan Citra Digital Untuk Identifikasi Objek Menggunakan Metode Hierarchical Agglomerative Clustering,” vol. 10, no. 2, pp. 148–156, 2021.

[7] A. R. Putri, “Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya,” JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform., vol. 1, no. 01, pp. 1–6, 2016, doi: 10.29100/jipi.v1i01.18.

[8] B. Sit, M. Iqbal Quraishi, and P. Student, “A Review Paper on Hough Transform and it’s Applications in Image Processing,” Int. J. Innov. Res. Sci. Eng. Technol. (An ISO, vol. 3297, p. 13, 2007, [Online]. Available: www.ijirset.com.

[9] V. Georgieva, P. Petrov, and A. Mihaylova, “GUI for Circular and Elliptic Objects Detection in Digital Images GUI für kreisförmigen und ellipsenförmigen Objektserkennung in den,” vol. 1, pp. 7–10, 2017.

[10] R. M. Putra, R. D. Puriyanto, K. Uad, and J. Ring, “Sistem Deteksi dan Pelacakan Bola dengan Metode Hough circle Transform Menggunakan Kamera Omnidirectional pada Robot Sepak Bola Beroda,” vol. 3, no. 3, pp. 176–184, 2021, doi: 10.12928/biste.v3i3.4786.

[11] R. A. Rizal, I. S. Girsang, and S. A. Prasetiyo, “Klasifikasi Wajah Menggunakan Support Vector Machine (SVM),” REMIK (Riset dan E-Jurnal Manaj. Inform. Komputer), vol. 3, no.2,p.1,2019,doi: 10.33395/remik.v3i2.10080.

[12] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[13] W. S. Maulsby, “Getting in News”, in Mondry, 2008, pp. 132-133

[14] A. Z. Arifin, and A. N. Setiono, ”Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering”, Institut Teknologi sepuluh November (ITS) Surabaya. http://mail.itssby.edu/~agusza/SITIAKlasifikasiEvent.pdf.

[15] V. A. Gunawan and L. S. A. Putra, “Comparison of American Sign Language Use Identification using Multi-Class SVM Classification,Backpropagation Neural Network, K - Nearest Neighbor and Naive Bayes,” Teknik, vol. 42, no. 2, pp. 137–148, 2021, doi:10.14710/teknik.v42i2.36929.

[16] J. Jumadi and D. Sartika, “Pengolahan Citra Digital Untuk Identifikasi Objek Menggunakan Metode Hierarchical Agglomerative Clustering,”vol. 10, no. 2, pp. 148–156, 2021.

[17] A. R. Putri, “Pengolahan Citra Dengan Menggunakan Web Cam Pada Kendaraan Bergerak Di Jalan Raya,” JIPI (Jurnal Ilm. Penelit. Dan Pembelajaran Inform., vol. 1, no. 01, pp. 1–6, 2016, doi: 10.29100/jipi.v1i01.18.

[18] B. Sit, M. Iqbal Quraishi, and P. Student, “A Review Paper on Hough Transform and it’s Applications in Image Processing,” Int. J. Innov.Res. Sci. Eng. Technol. (An ISO, vol. 3297, p. 13, 2007, [Online]. Available: www.ijirset.com.

[19] V. Georgieva, P. Petrov, and A. Mihaylova, “GUI for Circular and Elliptic Objects Detection in Digital Images GUI für kreisförmigen und ellipsenförmigen Objektserkennung in den,” vol. 1, pp. 7–10, 2017.

[20] R. M. Putra, R. D. Puriyanto, K. Uad, and J. Ring, “Sistem Deteksi dan Pelacakan Bola dengan Metode Hough circle Transform Menggunakan Kamera Omnidirectional pada Robot Sepak Bola Beroda,” vol. 3, no. 3, pp. 176–184, 2021, doi: 10.12928/biste.v3i3.4786.

[21] R. A. Rizal, I. S. Girsang, and S. A. Prasetiyo, “Klasifikasi Wajah Menggunakan Support Vector Machine (SVM),” REMIK (Riset dan EJurnal Manaj. Inform. Komputer), vol. 3, no. 2, p. 1, 2019, doi: 10.33395/remik.v3i2.10080.

[22] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” Int. J. Data Min. Knowl. Manag. Process, vol. 5,no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[23] M. F. Rahman, M. I. Darmawidjadja, and D. Alamsah, “kalsifikasi untuk Diagnosa Diabetes Menggunakan Metode Bayesian Regularization Neural Network (RBNN),” Inform., vol. 11, no. 1, pp. 36-45, 2017.

LAMPIRAN A

```
#otsu 2
# import required module
import os
import matplotlib as plt
import matplotlib.pyplot as plt
import numpy as np
import cv2
from skimage import data
from skimage.filters import threshold_multiotsu
import csv
matplotlib.rcParams['font.size'] = 9
# Setting the font size for all plots.
def createData(imageName, e_lines, e_circles, otsuResult, show=True):
  # The input image.
  image = cv2.imread(imageName,0)

  # Applying multi-Otsu threshold for the default value, generating
  # three classes.
  # thresholds = threshold_otsu(image)
  ret,th =
cv2.threshold(image,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
  ret, thresh1 = cv2.threshold(image, 120, 255, cv2.THRESH_BINARY +
                               cv2.THRESH_OTSU)

  # print(ret)
  # print(th)
  ret0 = [ret]
  # Using the threshold values, we generate the three regions.
  regions = np.digitize(image, bins=ret0)
```

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(28,28))

# Plotting the original image.
ax[0][0].imshow(image, cmap='gray')
ax[0][0].set_title('Original')
ax[0][0].axis('off')

# Plotting the histogram and the two thresholds obtained from
# multi-Otsu.
ax[0][1].hist(image.ravel(), bins=255)
ax[0][1].set_title('Histogram')
for thresh in ret0:
    ax[0][1].axvline(thresh, color='r')

# Plotting the Multi Otsu result.
ax[1][0].imshow(thresh1,cmap='gray')
ax[1][0].set_title('Otsu result')
ax[1][0].axis('off')

ax[1][1].hist(thresh1.ravel(), bins=255)
ax[1][1].set_title('Histogram Otsu')
for thresh1 in ret0:
    ax[1][1].axvline(thresh1, color='r')

plt.subplots_adjust()
if show == False:
   plt.close()
else :
  plt.show()
fig.savefig(otsuResult, dpi=300)
```

```
#deteksi garis
RGB = cv2.imread(imageName)

# Convert the img to grayscale
gray = cv2.cvtColor(RGB, cv2.COLOR_BGR2GRAY)

# Apply edge detection method on the image
edges = cv2.Canny(gray, 50, 150, apertureSize=3)



# This returns an array of r and theta values
lines = cv2.HoughLines(edges, 1, np.pi/18, 200)
# print(lines)
# The below for loop runs till r and theta values
# are in the range of the 2d array
backtorgb = cv2.cvtColor(image,cv2.COLOR_GRAY2RGB)
lines_pos = []
if lines is not None:
  for r_theta in lines:
      arr = np.array(r_theta[0], dtype=np.float64)
      r, theta = arr
      # Stores the value of cos(theta) in a
      a = np.cos(theta)

      # Stores the value of sin(theta) in b
      b = np.sin(theta)

      # x0 stores the value rcos(theta)
      x0 = a*r
```

```
        # y0 stores the value rsin(theta)
        y0 = b*r

        # x1 stores the rounded off value of (rcos(theta)-1000sin(theta))
        x1 = int(x0 + 1000*(-b))

        # y1 stores the rounded off value of (rsin(theta)+1000cos(theta))
        y1 = int(y0 + 1000*(a))

        # x2 stores the rounded off value of (rcos(theta)+1000sin(theta))
        x2 = int(x0 - 1000*(-b))

        # y2 stores the rounded off value of (rsin(theta)-1000cos(theta))
        y2 = int(y0 - 1000*(a))
        distance = np.linalg.norm([x1 - x2, y1 - y2])
        lines_pos_data = [x1,y1,x2,y2,distance]
        lines_pos.append(lines_pos_data)

        # cv2.line draws a line in img from the point(x1,y1) to (x2,y2).
        # (0,0,255) denotes the colour of the line to be
        # drawn. In this case, it is red.

        cv2.line(backtorgb, (x1, y1), (x2, y2), (0, 0, 255), 2)

# All the changes made in the input image are finally
# written on a new image houghlines.jpg

cv2.imwrite("linesDetected.jpg", backtorgb)  #not necessary at the moment
# Ll=len(lines)
```

```
# mendeteksi lingkaran
# Blur using 3 * 3 kernel.
gray_blurred = cv2.blur(gray, (3, 3))

# Apply Hough transform on the blurred image.
detected_circles = cv2.HoughCircles(gray_blurred,
                   cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
               param2 = 30, minRadius = 1, maxRadius = 40)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

  # for pt in detected_circles[0, :]:
    #     a, b, r = pt[0], pt[1], pt[2]

        # Draw the circumference of the circle.
    #    cv2.circle(img, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
      #  cv2.circle(img, (a, b), 1, (0, 0, 255), 3)
      # cv2.imshow("Detected Circle", img)
        #cv2.waitKey(0)



#Deteksi lingkaran

# Read image.
img = cv2.imread(imageName)
```

```
# Convert to grayscale.
g = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur using 3 * 3 kernel.
gb = cv2.blur(g, (3, 3))

# Apply Hough transform on the blurred image.
dc = cv2.HoughCircles(gb,
                cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
          param2 = 30, minRadius = 1, maxRadius = 40)
# print("Ini lingkaran", dc[0])
# print("Garis" , lines)

# export 'lines' variable to csv
# export 'dc' variable to csv

# data lingkaran
data_circles = [['x', 'y', 'r']]
for data in dc[0]:
  data_circles.append(data)



# name of csv file
filename = e_circles

# writing to csv file
with open(filename, 'w', newline='') as csvfile:
    # creating a csv writer object
    csvwriter = csv.writer(csvfile)
```

```
      # writing the data rows
      csvwriter.writerows(data_circles)

  # data garis
  data_lines = [['a', 'b','x1','y1','x2','y2','d']]
  if lines is not None:
    for i in range(len(lines)):
      dummy = [lines[i][0][0], lines[i][0][1]] + lines_pos[i]
      data_lines.append(dummy)
    # name of csv file
    filename = e_lines
    # writing to csv file
    with open(filename, 'w', newline='') as csvfile:
        # creating a csv writer object
        csvwriter = csv.writer(csvfile)
        # writing the data rows
        csvwriter.writerows(data_lines)
  return [ret, th, data_circles, data_lines]

def extractData(data, kategori):
  # Extract black and white value
  black = 0
  for subdata in data[1]:
    black += np.count_nonzero(subdata == 0)
  white = 0
  for subdata in data[1]:
    white += np.count_nonzero(subdata == 255)
  # Batas otsu
  batas_otsu = data[0]
  # print(black, white, batas_otsu)
  # Extract circles information
```

```
  jumlah_lingkaran =  len(data[2])-1
  # Sum all radius to get average
  sum_radius = round(sum([x[2] for x in data[2] if x[2] != 'r']), 2)
  average_radius = round(sum_radius/jumlah_lingkaran,2)
  # print(jumlah_lingkaran, sum_radius, average_radius)
  # Extract lines information
  jumlah_lines = len(data[3])-1
  # Sum all radius to get average
  sum_lines = round(sum([x[6] for x in data[3] if x[6] != 'd']), 2)
  try:
    average_lines = round(sum_lines/jumlah_lines,2)
  except:
    average_lines = 0
  # print(jumlah_lines, sum_lines, average_lines)
  return [black, white, batas_otsu, jumlah_lingkaran, average_radius,
jumlah_lines, average_lines, kategori]

#SVM
# assign directory
directory = 'cdots'
result_dir = "result_" + directory
# To export result, just upload your image to files folder and click run.
# iterate over files in
# that directory
hasil = []
for filename in os.listdir(directory):
    f = os.path.join(directory, filename)
    # checking if it is a file
    if os.path.isfile(f):
        f_temp = os.path.join(result_dir, os.path.splitext(filename)[0])
```

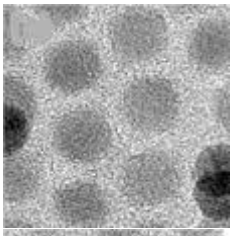
```
        # Set last parameter to True to show result, False to only save it. The value is
False by Default.
        data = createData(f, "{}-lines.csv".format(f_temp), "{}-
circles.csv".format(f_temp), "{}-otsu-result.jpg".format(f_temp), False)
        hasil_temp = extractData(data, directory)
        hasil.append(hasil_temp)
        print("{} done!".format(filename))

data_hasil = [['black',
'white','batas_otsu','n_lingkaran','average_radius','n_lines','average_line_length',
'kategori']]
for sub_data in hasil:
  data_hasil.append(sub_data)
with open('datasvm-cdots.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(data_hasil)

# Importing required libraries
from seaborn import load_dataset, pairplot
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
df = pd.read_csv('data.csv')
# df = df.drop(columns=['average_line_length'])
# Print the first 5 rows of the DataFrame
print(df.head())
#print(df.head())
# Dropping missing records
pairplot(df, hue='kategori')
```

```
plt.show()

X = df[['black', 'n_lingkaran']]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=100)

# Building and training our model
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)

import numpy as np
from seaborn import scatterplot
w = clf.coef_[0]
b = clf.intercept_[0]
x_visual = np.linspace(55,70) #Disesuaikan
y_visual = -(w[0] / w[1]) * x_visual - b / w[1]

scatterplot(data = X_train, x='black', y='n_lingkaran', hue=y_train) #Disesuaikan
plt.plot(x_visual, y_visual)
plt.show()
```

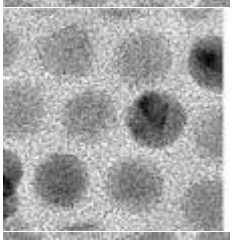
LAMPIRAN B

Kumpulan gambar TEM C-dots dan Non C-dots.

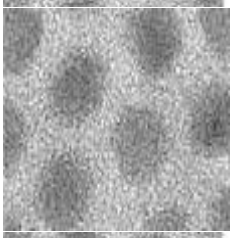
C dots 1



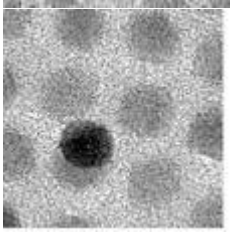
C dots 2



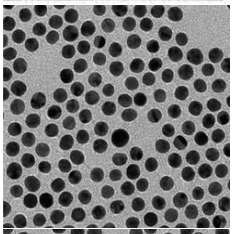
C dots 3



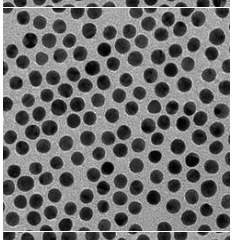
C dots 4



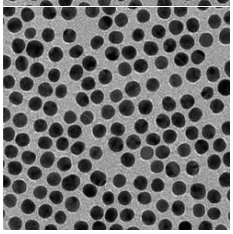
C dots 5



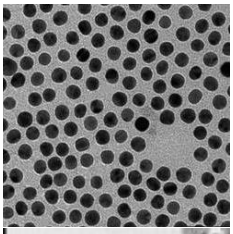
C dots 6



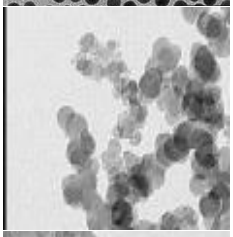
C dots 7



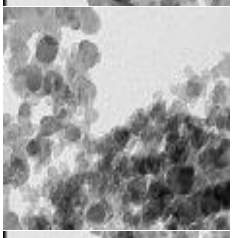
C dots 8



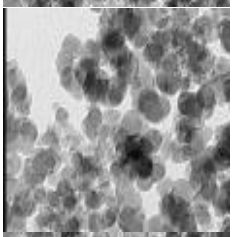
C dots 9



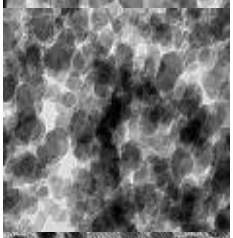
C dots 10



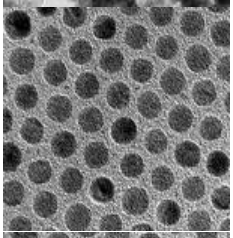
C dots 11



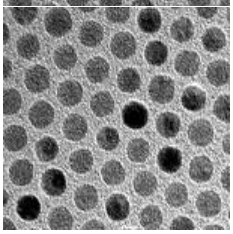
C dots 12



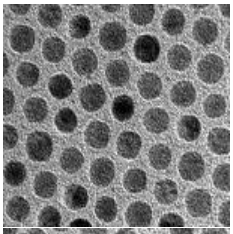
C dots 13



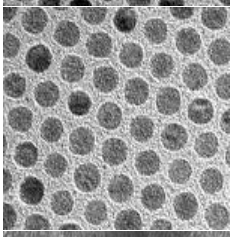
C dots 14



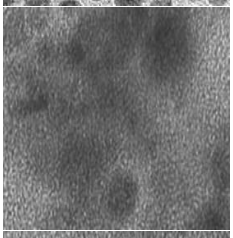
C dots 15



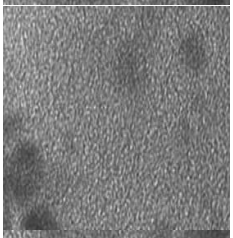
C dots 16



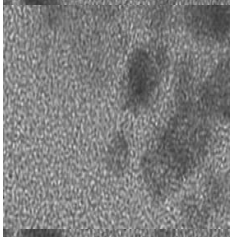
C dots 17



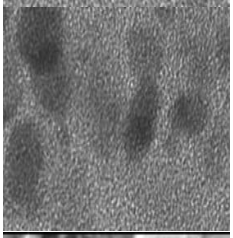
C dots 18



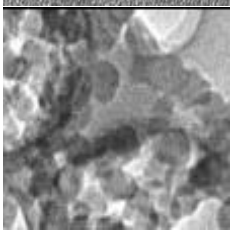
C dots 19



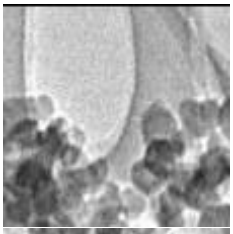
C dots 20



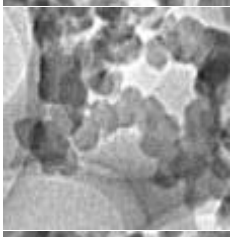
C dots 21



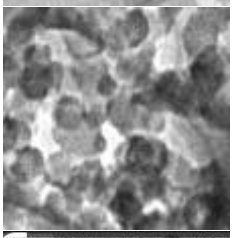
C dots 22



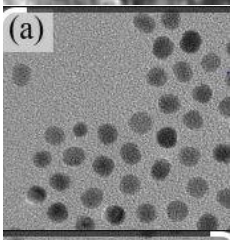
C dots 23



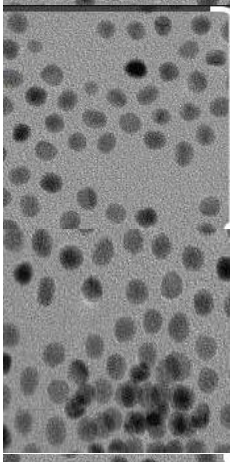
C dots 24



C dots 25

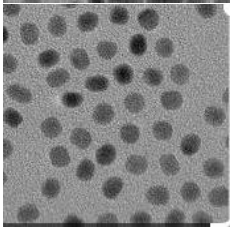


C dots 26

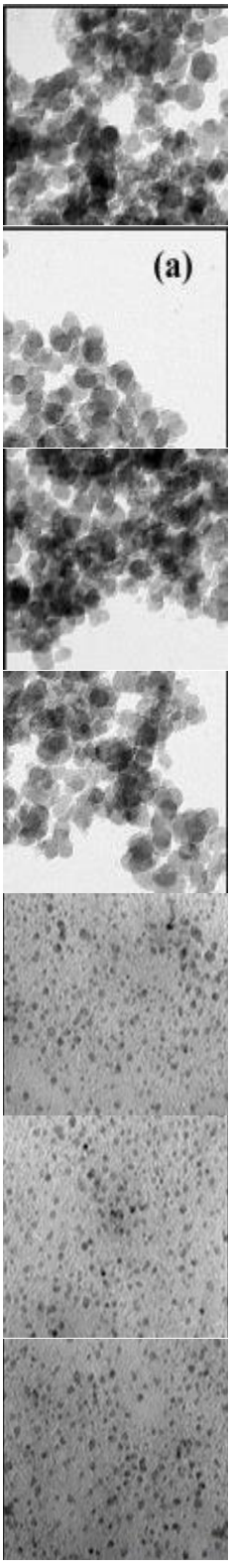


C dots 27

C dots 28



C dots 29



C dots 30

C dots 31

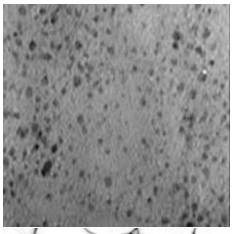
C dots 32

C dots 33

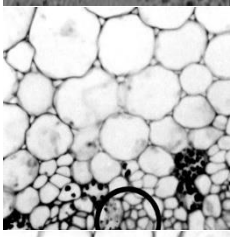
C dots 34

C dots 35

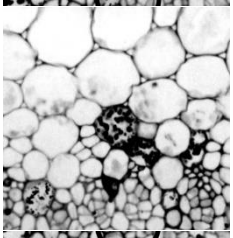
C dots 36



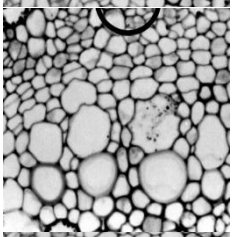
C dots 37



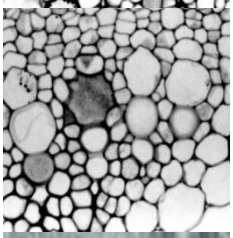
C dots 38



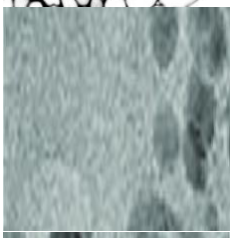
C dots 39



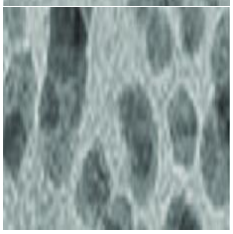
C dots 40



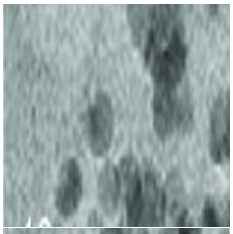
C dots 41



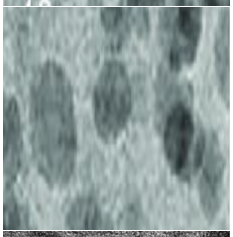
C dots 42



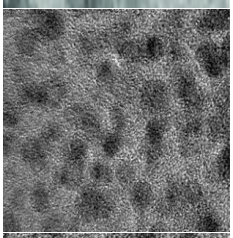
C dots 43



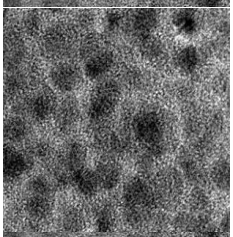
C dots 44



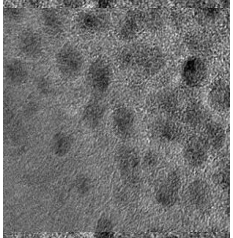
C dots 45



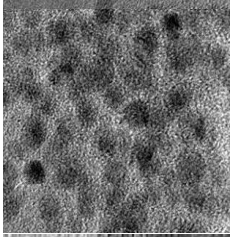
C dots 46



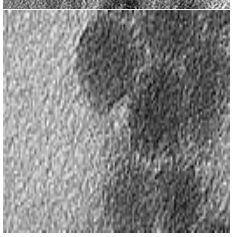
C dots 47



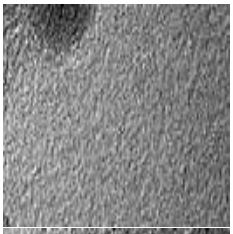
C dots 48



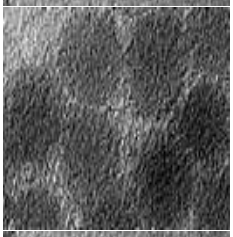
C dots 49



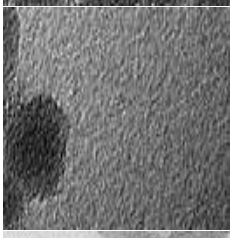
C dots 50



C dots 51



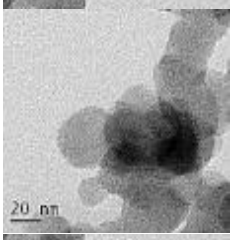
C dots 52



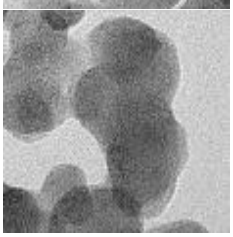
C dots 53

C dots 54

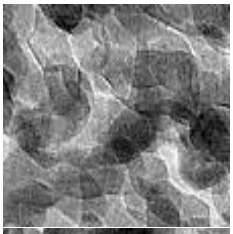
C dots 55



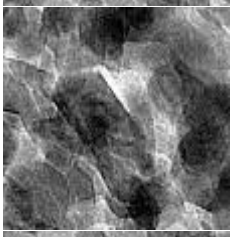
C dots 56



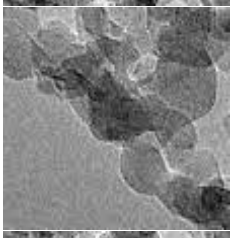
C dots 57



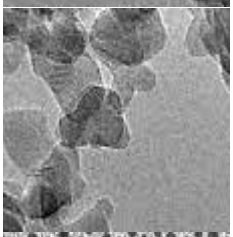
C dots 58



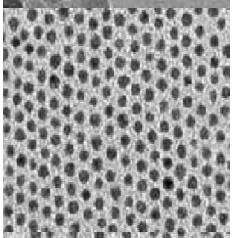
C dots 59



C dots 60

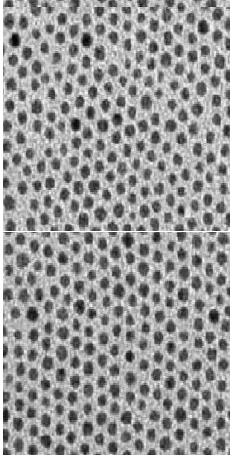


C dots 61

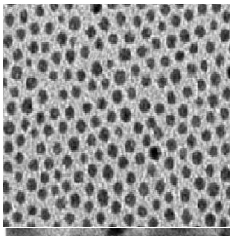


C dots 62

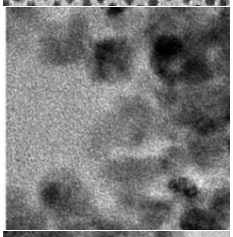
C dots 63



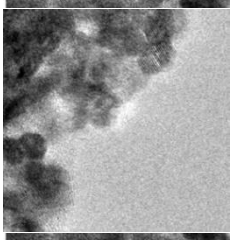
C dots 64



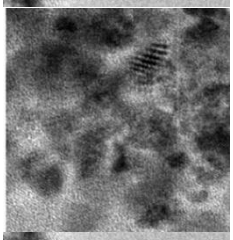
C dots 65



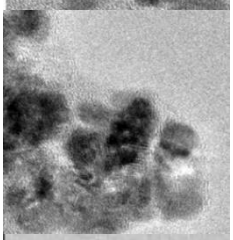
C dots 66



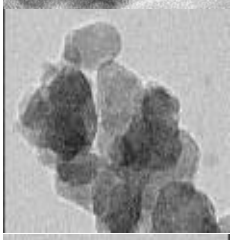
C dots 67



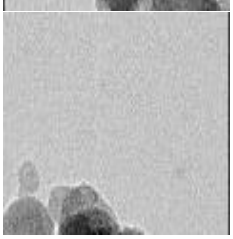
C dots 68



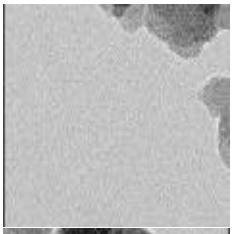
C dots 69



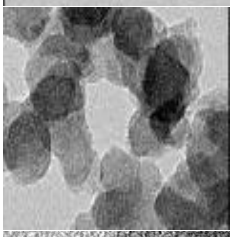
C dots 70



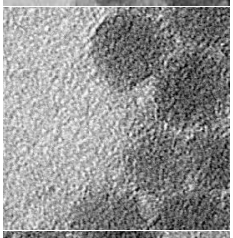
C dots 71



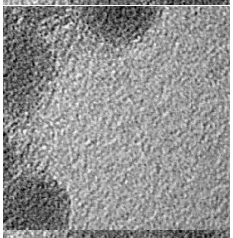
C dots 72



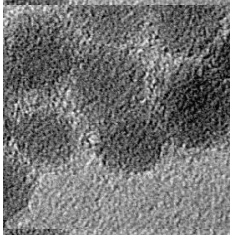
C dots 73



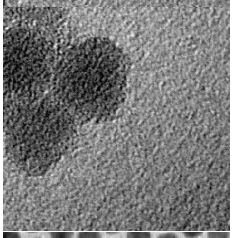
C dots 74



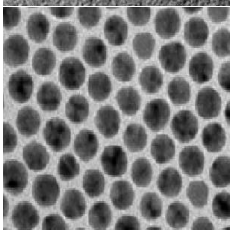
C dots 75



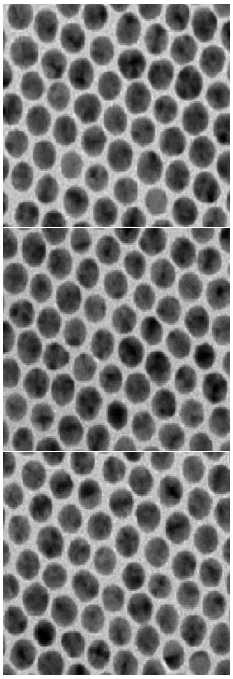
C dots 76



C dots 77



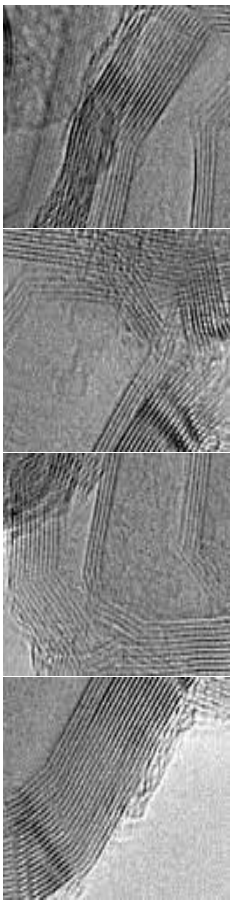
C dots 78



C dots 79

C dots 80

Non C dots 1

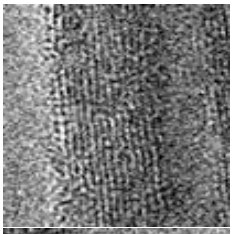


Non C dots 2

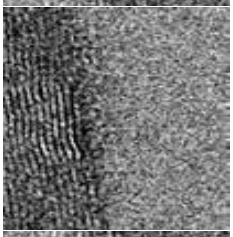
Non C dots 3

Non C dots 4

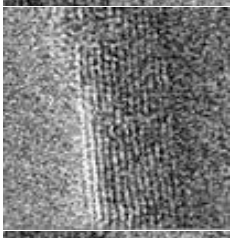
Non C dots 5



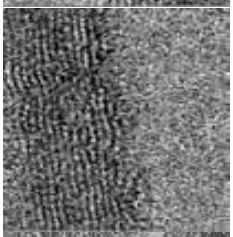
Non C dots 6



Non C dots 7



Non C dots 8

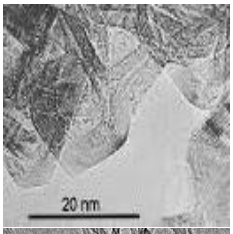


Non C dots 9

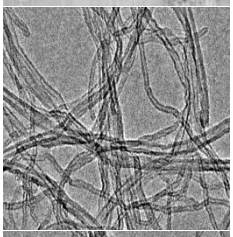
Non C dots 10

Non C dots 11

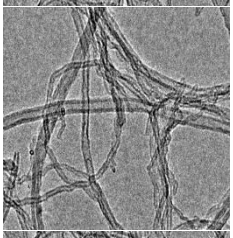
Non C dots 12



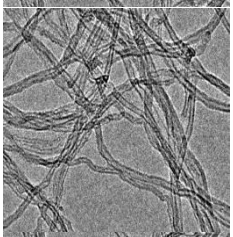
Non C dots 13



Non C dots 14

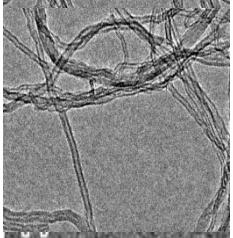


Non C dots 15

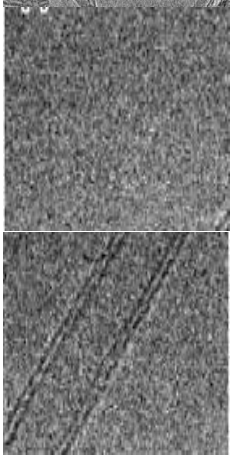


Non C dots 16

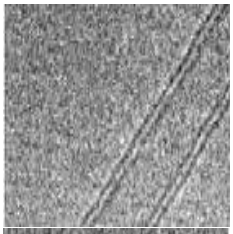
Non C dots 17



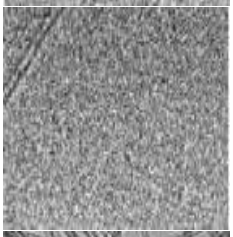
Non C dots 18



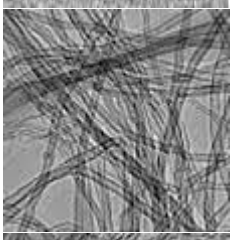
Non C dots 19



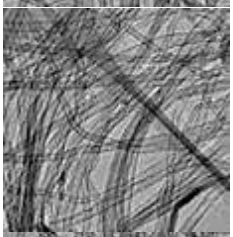
Non C dots 20



Non C dots 21

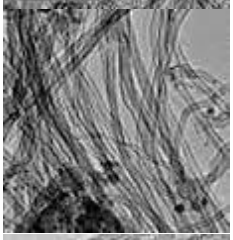


Non C dots 22

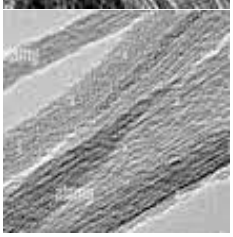


Non C dots 23

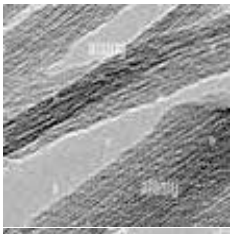
Non C dots 24



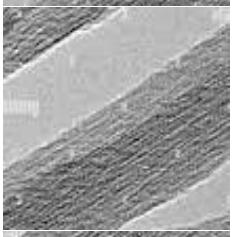
Non C dots 25



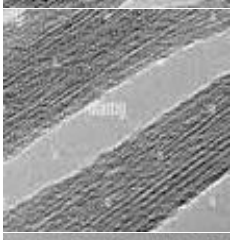
Non C dots 26



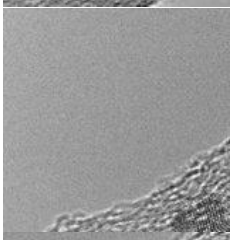
Non C dots 27



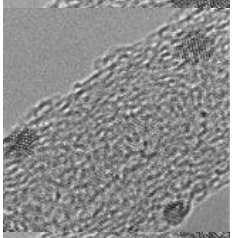
Non C dots 28



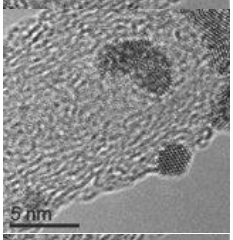
Non C dots 29



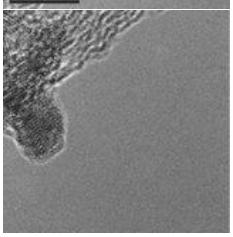
Non C dots 30



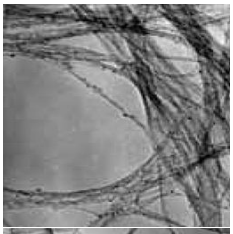
Non C dots 31



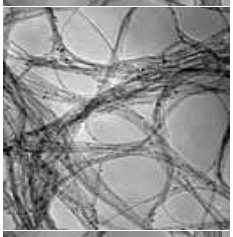
Non C dots 32



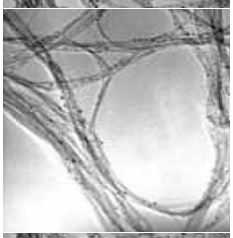
Non C dots 33



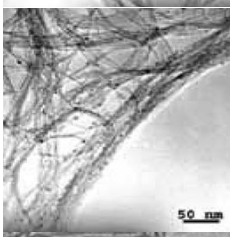
Non C dots 34



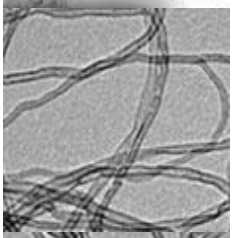
Non C dots 35



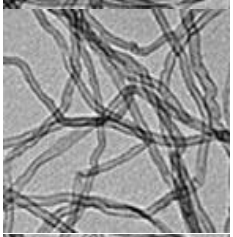
Non C dots 36



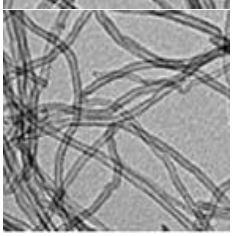
Non C dots 37



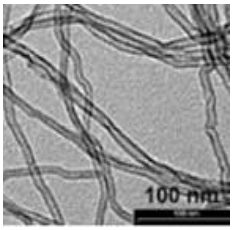
Non C dots 38



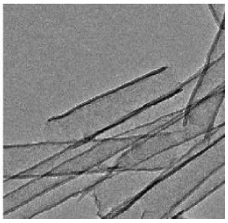
Non C dots 39



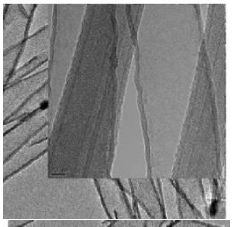
Non C dots 40



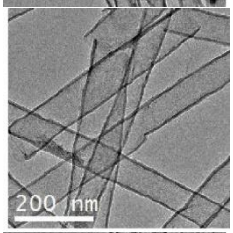
Non C dots 41



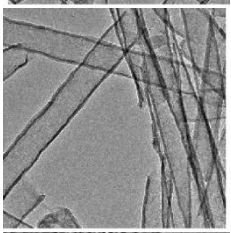
Non C dots 42



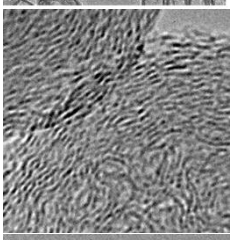
Non C dots 43



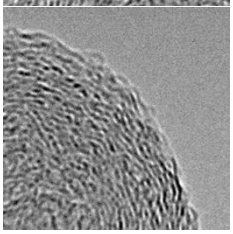
Non C dots 44



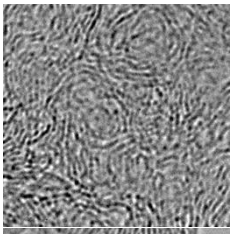
Non C dots 45



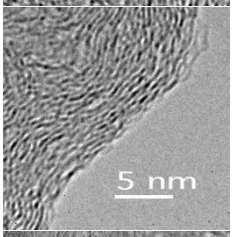
Non C dots 46



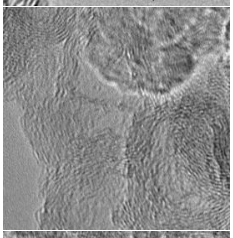
Non C dots 47



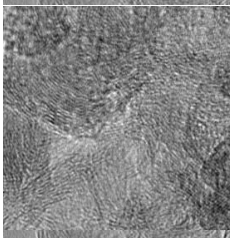
Non C dots 48



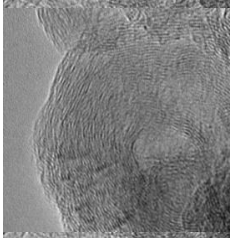
Non C dots 49



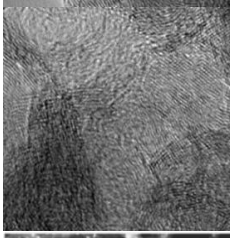
Non C dots 50



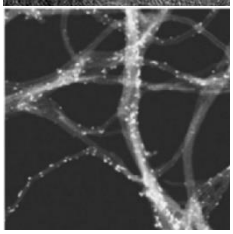
Non C dots 51



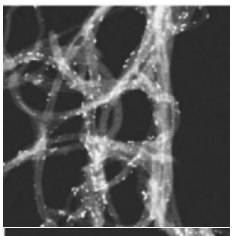
Non C dots 52



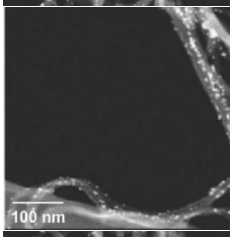
Non C dots 53



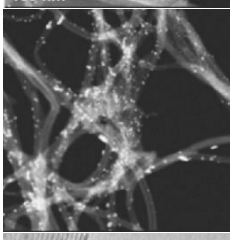
Non C dots 54



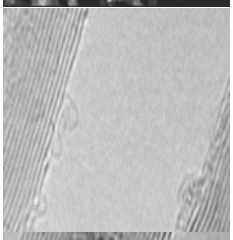
Non C dots 55



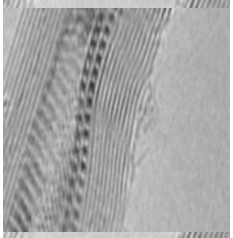
Non C dots 56



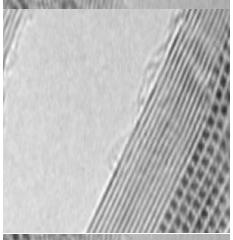
Non C dots 57



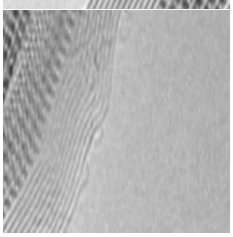
Non C dots 58



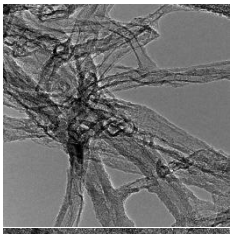
Non C dots 59



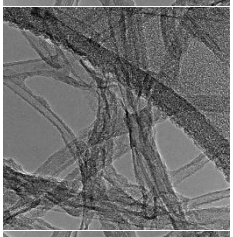
Non C dots 60



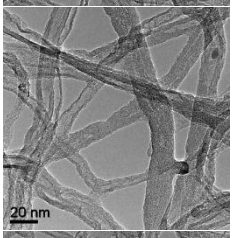
Non C dots 61



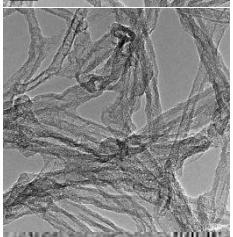
Non C dots 62



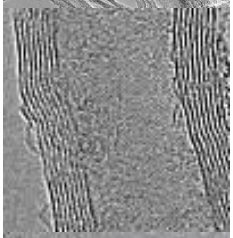
Non C dots 63



Non C dots 64

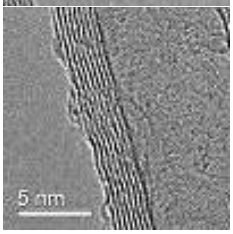


Non C dots 65

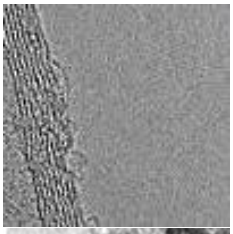


Non C dots 66

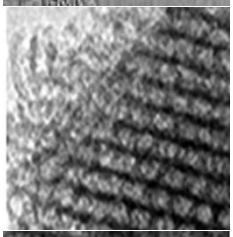
Non C dots 67



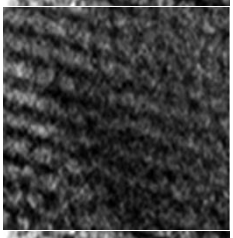
Non C dots 68



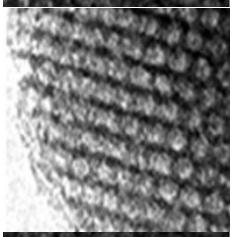
Non C dots 69



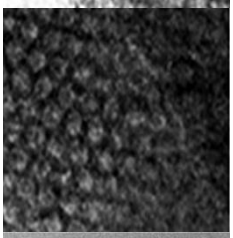
Non C dots 70



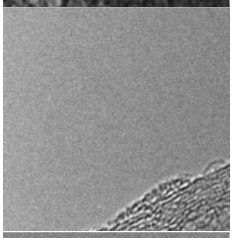
Non C dots 71



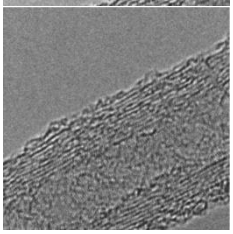
Non C dots 72



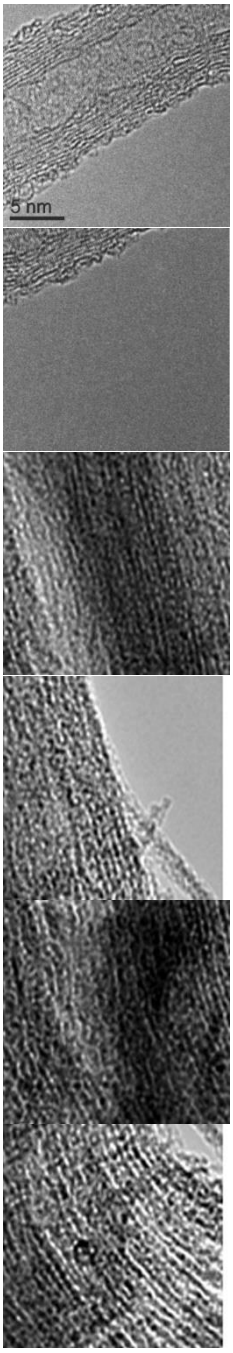
Non C dots 73



Non C dots 74



Non C dots 75



Non C dots 76

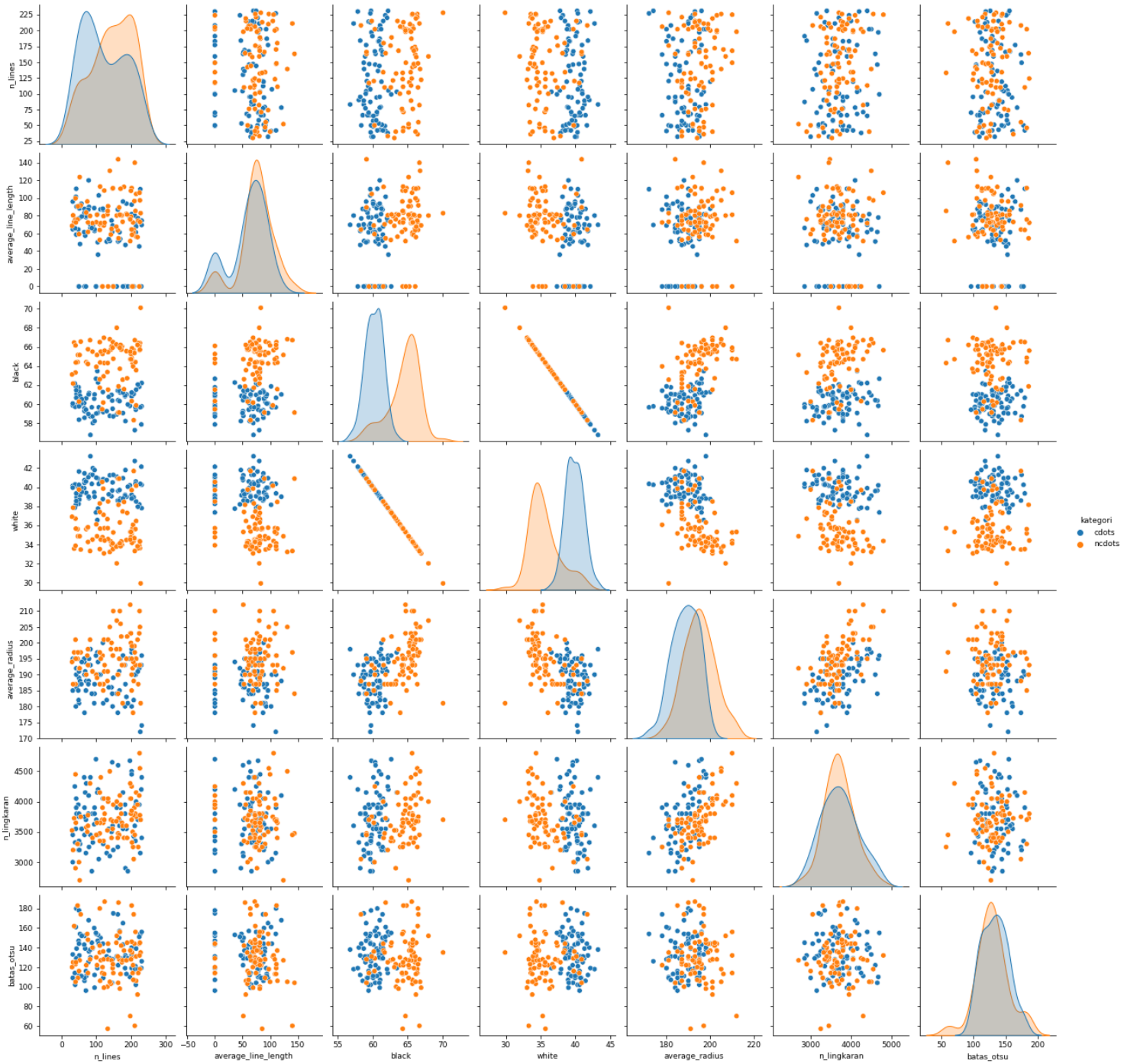
Non C dots 77

Non C dots 78

Non C dots 79

Non C dots 80

Lampiran C


n_lines
average_line_length
black
white
average_radius
n_lingkaran
batas_otsu
kategori
cdots
ncdots