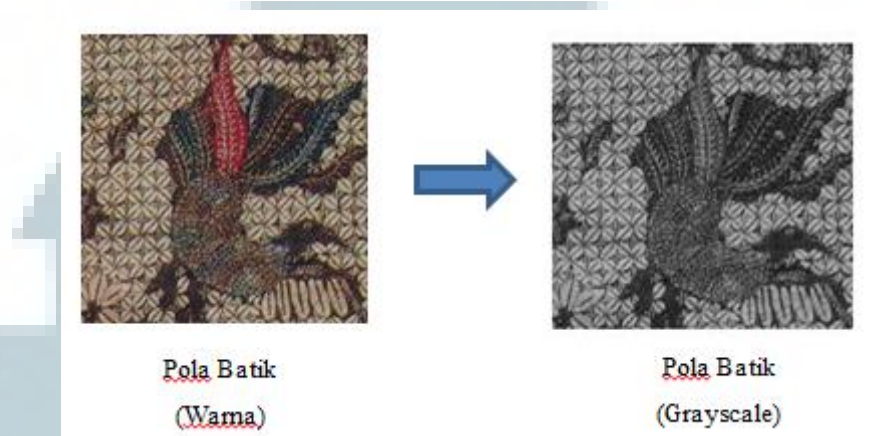
```
folder = 'C:\Users\ambonaz\Documents\SKRIPSI'; % mendeklarasi parameter yang berisi lokasi file yang akan dipakai
filepattern = imread(folder, 'sample*.png'); % membaca nama file yang akan dipakai
image_gray=rgb2gray(filepattern); % mengkonversi dari 'truecolor' atau warna asli pada gambar menjadi grayscale
imshow (image_gray) % menampilkan gambar yang sudah dikonversi menjadi grayscale
```

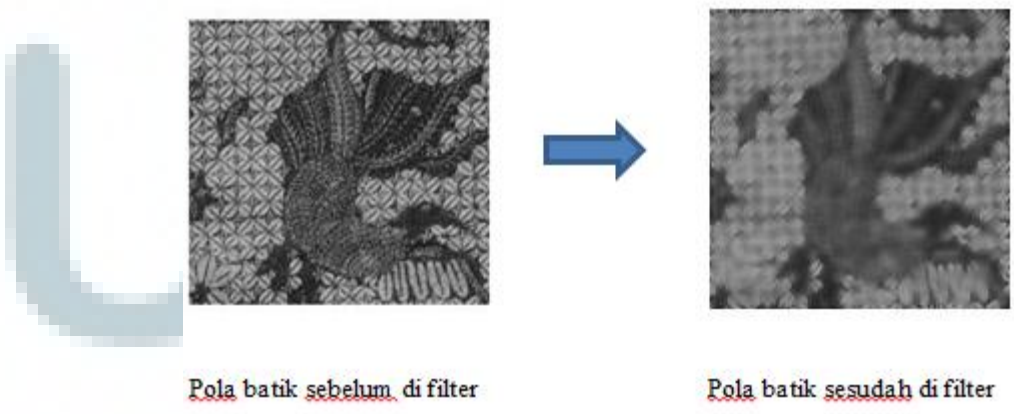
Gambar 4.2 Pseudocode cara kerja grayscale pada matlab



Gambar 4.3 Contoh pengaplikasian gray scale pada penelitian

4.1.4. Wiener *Filter*

Dalam proses *image processing*, terkadang terdapat masalah yang cukup besar yang ditimbulkan dan berasal dari gambar yang akan diproses. Untuk mengurangi efek dari masalah tersebut, penulis menggunakan Wiener *filter* yang bekerja dengan baik dalam mereduksi tingkat gangguan yang ditimbulkan, seperti yang dapat terlihat pada gambar 4.4



Gambar 4.4 Contoh pengaplikasian wiener filter pada penelitian

Untuk menggunakan Wiener *filter* pada penelitian ini, penulis menggunakan *syntax* `Filtered = wiener2 (l,[m,n]);` dimana 'Filtered' adalah variabel untuk hasil Wiener *filter* , 'l' adalah gambar yang ingin difiltrasi, 'm' adalah panjang filter yang akan diterapkan (dalam satuan *pixel*), dan 'n' adalah lebar filter yang akan diterapkan (dalam satuan *pixel*).

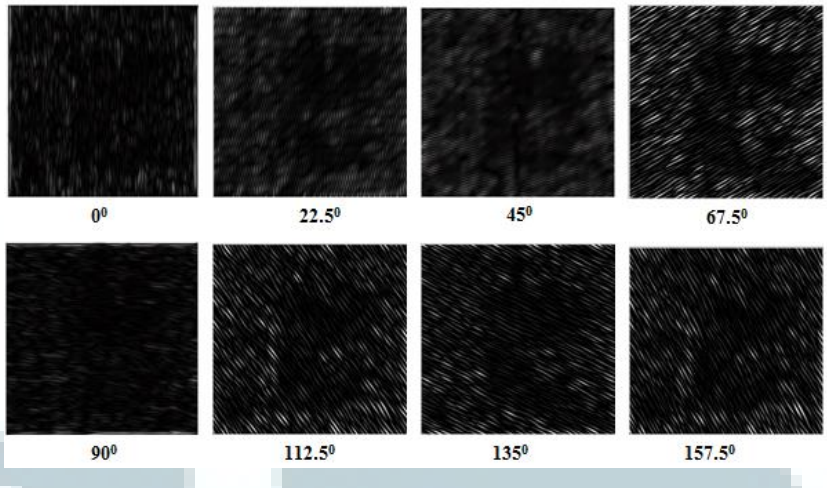
Untuk penggunaan teknik filtrasi pada penelitian ini penulis menggunakan ukuran sebesar 5x5 *pixel*. Sehingga penggunaan *syntax* dengan penerapan ukuran tersebut yaitu W = wiener2(I, [5 5]).

4.1.5. Gabor *Filter*

Untuk dapat mengekstraksi ciri khas dari nilai yang ada pada setiap sampel, maka yang pertama dilakukan adalah penyesuaian terhadap parameter dari Gabor *filter*. Pada tahap ini, penulis menentukan berapa banyak orientasi dan panjang gelombang (*wavelength*) yang harus dilakukan agar nilai yang dihasilkan dapat digunakan untuk proses selanjutnya.

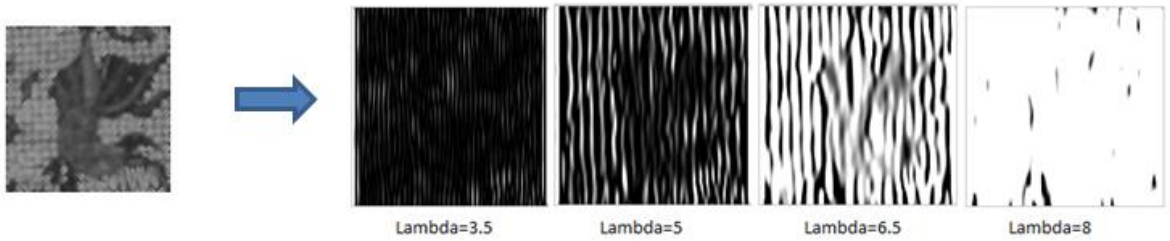
Dalam menentukan orientasi, penulis menggunakan delapan sudut yang dimana setiap pergeseran sudut tersebut akan bertambah sebanyak $22{,}5^{o}$

Alasan pengambilan sudut tersebut dikarenakan nilai yang dihasilkan lebih baik dari percobaan penulis sebelumnya yang menggunakan 4 sudut yaitu 0 , pi/4,pi/2,dan 3pi/4 yang menghasilkan akurasi sebesar 40%. Hasil dari pergeseran orientasi ini dapat terlihat pada gambar 4.5



Gambar 4.5 Orientasi pada gabor filter yang digunakan penulis

Pada langkah berikutnya, penulis menentukan panjang gelombang (*wavelength*). Parameter ini berfungsi sebagai penentu tebal atau tipisnya *filter*. Penulis menggunakan λ=3.5 sebagai nilai yang paling terkecil dan selanjutnya bertambah λ=1.5 pada setiap iterasi dengan batas atas λ=8. Contoh hasil dari parameter *wavelength* dapat terlihat pada gambar 4.5



Gambar 4.6 Panjang gelombang pada gabor yang digunakan penulis

Parameter orientasi dan panjang gelombang tersebut dimasukkan ke dalam fungsi Gabor dengan *source code* pada aplikasi Matlab seperti berikut ini

```
lambda=3.5;
gamma=0.3;
sigma=2.8;
psi=0;

sigma_x = sigma;
sigma_y = sigma/gamma;

nstds = 5;
xmax = max(abs(nstds*sigma_x*cos(theta)),abs(nstds*sigma_y*sin(theta)));
xmax = ceil(max(1,xmax));
ymax = max(abs(nstds*sigma_x*sin(theta)),abs(nstds*sigma_y*cos(theta)));
ymax = ceil(max(1,ymax));
xmin = -xmax; ymin = -ymax;
[x,y] = meshgrid(xmin:xmax,ymin:ymax);

x_theta=x*cos(theta)+y*sin(theta);
y_theta=-x*sin(theta)+y*cos(theta);

gb= exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*cos(2*pi/lambda*x_theta+psi);
```

Sehingga, gb=gabor(sigma,theta,lambda,psi,gamma), yang di mana nilai *default* dari *source code* gabor *filter* yang sada dapat, *sigma* = 2.8 , *theta* = 0 , *lambda* = 3.5, *psi* = 0, dan *gamma* = 0.3. Dari fungsi tersebut penulis mendapatkan hasil berupa nilai energi berupa formasi matriks dari setiap iterasi Gabor filter yang digunakan masing-masing menghasilkan nilai dengan ukuran 2500 *pixel* dikarenakan gambar berukuran 50x50 *pixel*.

Langkah selanjutnya adalah mencari nilai rata-rata dari satu gambar yang diiterasikan, penulis menjumlahkan kumpulan-kumpulan matriks tersebut dengan rumus sebagai berikut

$$x = \frac{k1 + k2 + k3 + \cdots + kN}{N}$$

Sehingga menghasilkan satu nilai matriks rata-rata dari keseluruhan parameter Gabor filter. Selanjutnya, nilai tersebut akan digunakan untuk proses selanjutnya yaitu pengonversian nilai matriks ke vektor.

4.1.3. *Matrix Normalization*

Seperti yang sudah dijelaskan pada bagian 2.6 normalisasi matrik dilakukan untuk mendapatkan jenis matrik *orthogonal* yang akan digunakan pada langkah berikutnya. Untuk menormalisasi matrik dalam aplikasi matlab , penulis menggunakan *syntax*

```
normalize = NW - mean(mean(NW));
```

Yang mana ‘normalize’ adalah sebagai variabel yang menyimpan hasil dari normalisasi , dan ‘NW’ adalah variable matrik yang akan dinormalisasikan.

4.1.4. *Matrix to Vector Conversion*

Sebelum melanjutkan ke proses PCA, penulis menggabungkan nilai energy rata-rata dari setiap matriks yang terdapat pada tiap gambar yang selanjutnya akan digabungkan. Untuk mengombinasikan nilai rata-rata tersebut, terlebih dahulu penulis mengonversikan matriks yang berukuran 50x50 menjadi vector dengan ukuran 1x2500, seperti yang dapat terlihat pada tabel 4.1 dengan menggunakan fungsi *syntax* `matriks = gambar(:)';`.

	Kolom 1	Kolom 2	Kolom ...	Kolom n
Baris 1	0.2916	0.3776	...	0.2567
Baris 2	0.2557	0.1151	...	0.1313
Baris 3	0.0805	0.1547	...	0.1224

Vector	Baris 1	Baris 2	Baris 3

Tabel 4.1 Ilustrasi konversi matriks 1x2500

Dari gambar 4.6 dapat terlihat, bahwa nilai tersebut “digabungkan” satu persatu sehingga menghasilkan matriks baru berukuran 21x2500 dimensi yang

berisi nilai rata-rata dari hasil setiap gambar yang telah difiltrasi menggunakan Gabor seperti yang dapat terlihat pada tabel 4.2 dengan *syntax*

```
NW = vertcat (NW , matriks);
```

Vector 1	Baris 1	Baris 2	Baris 3
Vector 2	Baris 1	Baris 2	Baris 3

Vector 1	Baris 1	Baris 2	Baris 3
Vector 2	Baris 1	Baris 2	Baris 3
Vector ...	Baris ...	Baris ...	Baris ...
Vector n	Baris n	Baris n	Baris n

Tabel 4.2 Ilustrasi konversi matriks 16x2500

Berikut adalah *pseudocode* pada proses *matrix to vector conversion*

```
for k = 1 : 21   %pengulangan sebanyak data yg digunakan
    gambar = a1 {k}; %menyimpan hasil dari proses sebelumnya (a1) pada variabel gambar yang berisi matriks berukuran 50x50
    matriks = gambar(:)'; %menjadikan matriks berukuran 1x2500

     if k == 1
        NW = matriks;
     else
        NW = vertcat (NW , matriks); %menggabungkan matrisk 1x2500 menjadi 21x2500

     end


end
```

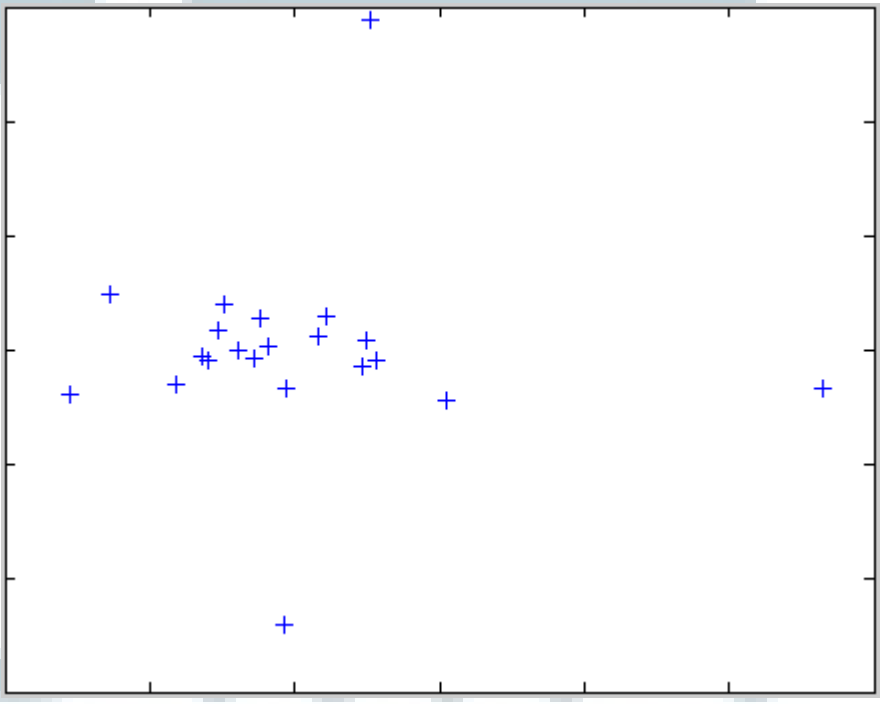
Gambar 4.7 Pseudocode matrix to vector conversion

Berdasarkan gambar 4.7 , untuk mengubah matriks menjadi 1x2500 menggunakan *syntax* **Matriks = gambar(:)';** dimana 'Matriks' adalah sebagai

variabel untuk menyimpan hasil dari matriks gambar yang sudah diubah orientasi bentuk matriksnya. Sedangkan untuk mengubah matriks menjadi 21x2500 menggunakan *syntax* `NW = vertcat(NW,matriks);` dimana 'NW' adalah sebagai variabel untuk menyimpan hasil dari matriks yang sudah berukuran 1x2500, yang kemudian digabungkan dengan menambah baris dari tiap iterasi yang dilakukan, sehingga di akhir iterasi akan menghasilkan matriks 21x2500.

4.1.5. Principal Component Analysis

Setelah menggabungkan beberapa vector, selanjutnya melakukan normalisasi matriks yang berisi vector, pada proses sebelumnya. Kemudian nilai normalisasi tersebut akan menjalani proses PCA, pada aplikasi MATLAB fungsi ini menggunakan *syntax* `[coeff score] = princomp(normalize);`. Dari hasil proses PCA tersebut, penulis hanya mengambil 2 dimensi dari 2500 dimensi untuk setiap nilai gambar, agar nilai dapat direpresentasikan kedalam bentuk grafik 2-dimensi. Seperti yang dapat terlihat pada gambar 4.8



Gambar 4.8 Representasi PCA

Pada gambar 4.8 hasil pemetaan PCA dari total 16 gambar yang dapat terlihat bahwa nilai energi yang dihasilkan terpencar dan tidak tersentralisasi dengan kata lain representasi nilai energi tersebut belum dapat dibedakan .

4.1.5. Support Vector Machine (SVM)

Untuk mengklasifikasikan pola gambar batik dari daerah Tanjung Bumi Bangkalan dengan pola gambar batik dari daerah lainnya secara otomatis pada saat *training* dan *testing*, penulis menggunakan SVM dua dimensi dari MATLAB. SVM ini bekerja dengan membuat *hyperplane* dari hasil atau matriks pada proses PCA.

Berdasarkan metode *K-Fold Cross Validation* yang dimana total jumlah sampel dibagi sejumlah *K* yang ditentukan, pada penelitian ini penulis, menggunakan 21 sampel yang terdiri dari 7 sampel dari daerah Tanjung Bumi Bangkalan dan 14 sampel dari dari daerah lainnya yang kemudian dibagi menjadi 7 *K* untuk penentuan data *testing*. Pada penentuan data *training* dan data *testing*, penulis melakukan pengacakan data dengan menggunakan *pseudocode* dibawah ini :

```
aa = [1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0]%label gambar sebelum diacak

random = randswap(1:21); %mengacak angka mulai dari angka 1 sampai 21
for i=1:size(random,1) %mengambil isi dari proses sebelumnya berupa matriks kolom pertama
%dari gambar yang sesuai dengan urutan angka yang telah diacak

for j=1:size(random,2) %mengambil isi dari proses sebelumnya berupa matriks kolom kedua
%dari gambar yang sesuai dengan urutan angka yang telah diacak

pointer = random(i,j) %memasukan hasil dari variabel i dan j yg sudah sesuai dengan urutan angka yang
%telah diacak

 databaru=data2d(pointer,1:2); %memasukan hasil dari pointer yang berupa gambar yg sudah sesuai dengan urutan
 %yang telah diacak kedalam variabel databaru
 datanew(j,1:2)=databaru;% mengambil gambar beserta nilai yg terdapat pada tiap gambar
 labelnew(j)=aa(pointer);% mengambil label tiap gambar sesuai urutan yang sudah dicak
 end;
end;
```

Gambar 4.9 Pseudocode mengacak data sampel

Dari gambar 4.9 , menghasilkan urutan gambar dan urutan label yang baru. Yang dimana sebelumnya mempunyai urutan berikut:

Gambar = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]

Label = [1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0]

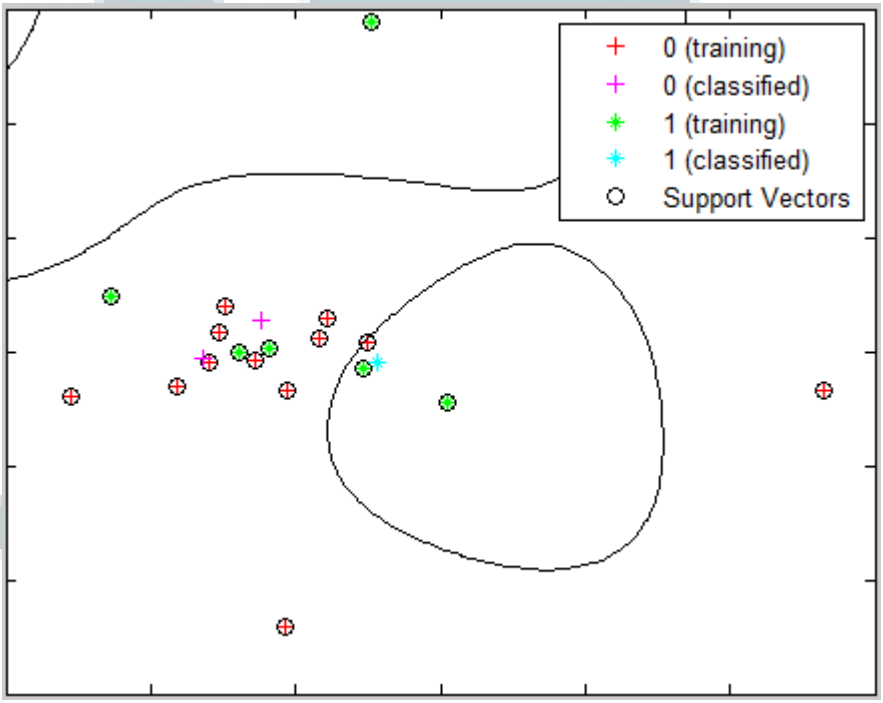
Menjadi

Gambar = [4,19,1,12,3,8,16,21,9,15,17,10,18,5,20,14,7,13,6,11,2]

Label = [0,0,1,0,1,0,0,0,1,0,0,0,0,1,0,0,1,1,0,1,0]

Proses *training* pada program mengambil hasil dari SVM pada K-1. Penulis melabelkan gambar dengan angka 1 untuk sampel gambar yang berasal dari daerah Tanjung Bumi Bangkalan dan 0 untuk sampel dari daerah lainnya.

Berdasarkan metode *training* "K-Fold" dengan menggunakan *kernel* Gaussian (rbf) dapat menghasilkan akurasi klasifikasi yang baik yaitu sebesar 100% pada pola batik Tanjung Bumi Bangkalan dengan pola batik daerah lainnya dengan memakan waktu 0.438018 detik untuk 14 gambar. Seperti yang dapat terlihat pada gambar 4.7



Gambar 4.10 Representasi SVM

Pada gambar 4.10 , penulis sudah dapat membedakan pola batik dari daerah Tanjung Bumi Bangkalan dan dari daerah lainnya. Yang dimana titik hijau berarti batik dari daerah Tanjung Bumi Bangkalan dan titik merah adalah batik dari daerah lainnya, sedangkan titik dengan lambang '+' dan berwarna jingga adalah hasil dari klasifikasi yang terbaca sebagai batik daerah lain sedangkan titik degan lambing '*' dan berwarna biru muda adalah hasil dari klasifikasi yang terbaca sebagai batik dari daerah Tanjung Bumi Bangkalan.

4.1.6. Menghitung Akurasi

Agar dapat mengukur akurasi secara matematis, penulis menggunakan hasil dari perbandingan yang didapat dari *Support Vector Machine* (SVM). Seperti yang dapat terlihat pada tabel dibawah ini

	Ground Truth	***Prediction***
True Positive	1	1
True Negative	0	0
False Positive	0	1
False Negative	1	0

Tabel 4.3 Tabel akurasi

Terlihat pada 4.3, bahwa dapat dikatakan sebagai :

- *True Positive* apabila *ground truth* dan *prediction* sama-sama bernilai 1
- *True Negative* apabila *ground truth* dan *prediction* sama-sama bernilai 0
- *False Positive* apabila *ground truth* bernilai 0 dan *prediction* bernilai 1
- *False Negative* apabila *ground truth* bernilai 0 dan *prediction* bernilai 0

Sedangkan untuk rumus yang penulis gunakan untuk memperoleh akurasi dari gambar, yaitu

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}\ X\ 100$$

Sehingga untuk pola batik saat proses *training* dapat dilihat dengan tabel seperti dibawah ini

	Ground Truth	*Prediction*	*Result*
 Sample4.png	0	0	*True Negative*
 Sample19.png	0	0	*True Negative*
 Sample1.png	1	1	*True Positive*

Tabel 4.4 Tabel hasil testing pada training

Dari tabel tersebut menghasilkan 1 true positive dan 2 true negative sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{1+2}{1+2+0+0} \; X \; 100 = 100$$

4.2 Testing Gambar

4.2.1 Image Processing

Setelah melakukan proses *training*, langkah selanjutnya yaitu melakukan proses *testing* gambar. Yang dimana pada proses ini akan langsung diklasifikasikan terhadap hasil dari model SVM pada proses *training*.

Untuk proses gambar secara keseluruhan persis dengan proses *training* (lihat 4.1.2 Analisa Proses Batik) dengan sedikit perbedaan yaitu

1. *Image usage*

   Gambar yang digunakan dalam tahap ini yaitu keseluruhannya merupakan bahan *testing* sehingga tidak ada gambar yang diklasifikasikan untuk *training*.

2. SVM *classify*

   Merupakan model SVM yang akan digunakan sebagai *tools* dalam mengklasifikasikan gambar apakah termasuk batik dari daerah Tanjung Bumi Bangkalan atau batik dari daerah lain.

4.2.2 Accuracy Test

Untuk memenuhi poin ke-2 tujuan penelitian, maka penulis berkewajiban untuk menghitung akurasi, dikarenakan ambang batas yang penulis tentukan minimum 80%, maka hasil yang diharapkan dapat melampaui target tersebut. Penghitungan akurasi menggunakan metode yang sama persis dengan proses *training* sebelumnya (lihat hal. 46).

Hasil pengetesan pada sampel *Testing* berdasarkan *ground truth* dan perhitungan *true positive* serta *false positive* dengan menggunakan SVM proses *training* menghasilkan akurasi seperti yang dapat terlihat pada tabel berikut ini

K-2			
	Ground Truth	*Prediction*	*Result*

 Sample12.png	0	0	*True Negative*
 Sample3.png	1	1	*True Positive*
 Sample8.png	0	1	*False Positive*

Tabel 4.5 tabel hasil testing K-2

Dari tabel tersebut menghasilkan 1 true positive ,1 true negative dan 1 false positive sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{1+1}{1+1+1+0} X\ 100 = 66.67$$

Lama proses *testing K-2* = 0.413860 detik

K-3			
	Ground Truth	*Prediction*	*Result*

 Sample16.png	0	0	*True Negative*
 Sample21.png	0	1	*False Positive*
 Sample9.png	1	1	*True Positive*

Tabel 4.6 tabel hasil testing K-3

Dari tabel tersebut menghasilkan 1 true positive , 1 true negative dan 1 false positive sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{1+1}{1+1+1+0} X\ 100 = 66.67$$

Lama proses *testing K-3* = 0.391576 detik

K-4			
	Ground Truth	*Prediction*	*Result*

 Sample15.png	0	0	*True Negative*
 Sample17.png	0	0	*True Negative*
 Sample10.png	0	1	*False Positive*

Tabel 4.7 tabel hasil testing K-4

Dari tabel tersebut menghasilkan 1 false positive dan 2 true negative sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{0+2}{0+2+1+0} X\ 100 = 66.67$$

Lama proses *testing K-4* = 0.373807 detik

K-5			
	Ground Truth	*Prediction*	*Result*

 Sample18.png	0	0	*True Negative*
 Sample5.png	1	0	*False Negative*
 Sample20.png	0	0	*True Negative*

Tabel 4.8 tabel hasil testing K-5

Dari tabel tersebut menghasilkan 1 false negative dan 2 true negative sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{0+2}{0+2+0+1} X\ 100 = 66.67$$

Lama proses *testing K-5* = 0.389391 detik

K-6			
	Ground Truth	*Prediction*	*Result*

 Sample14.png	0	0	*True Negative*
 Sample7.png	1	0	*False Negative*
 Sample13.png	1	0	*False Negative*

Tabel 4.9 tabel hasil testing K-6

Dari tabel tersebut menghasilkan 2 false negative dan 1 true negative sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{0+1}{0+1+0+2} X\ 100 = 33.33$$

Lama proses *testing K-6* = 0.389395 detik

K-7			
	Ground Truth	*Prediction*	*Result*

 Sample6.png	0	0	*True Negative*
 Sample11.png	1	1	*True positive*
 Sample2.png	0	0	*True Negative*

Tabel 4.10 tabel hasil testing K-7

Dari tabel tersebut menghasilkan 1 True Positive dan 2 true negative sehingga jika dimasukkan ke dalam rumus akurasi sebelumnya, maka

$$Accuracy = \frac{1+2}{1+2+0+0} \; X \; 100 = 100$$

Lama proses *testing K-7* = 0.390190 detik

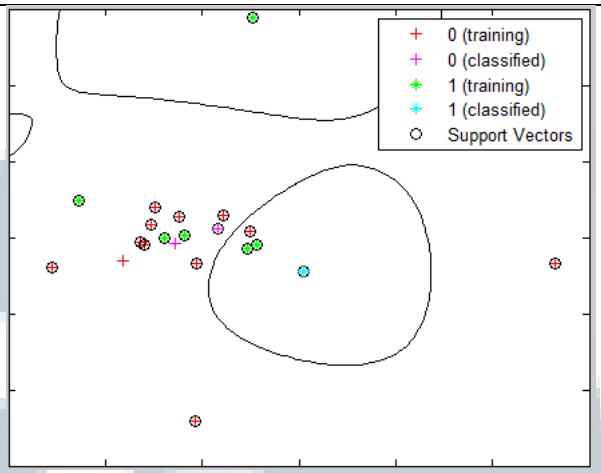
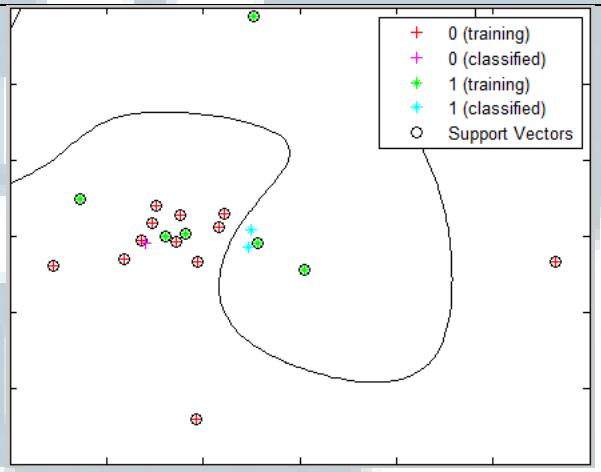
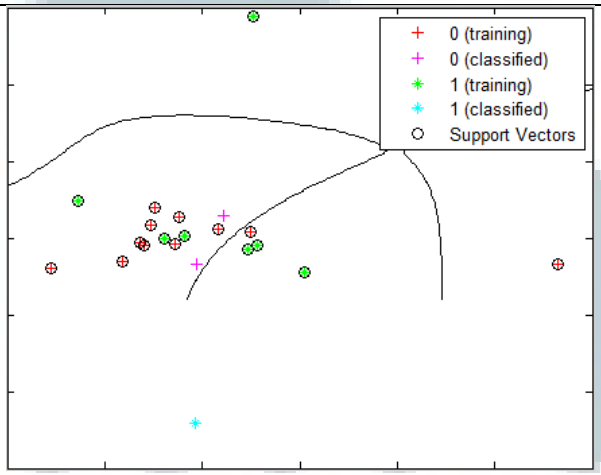
Dari beberapa testing tersebut, dapat menghasilkan rata-rata sebagai berikut:

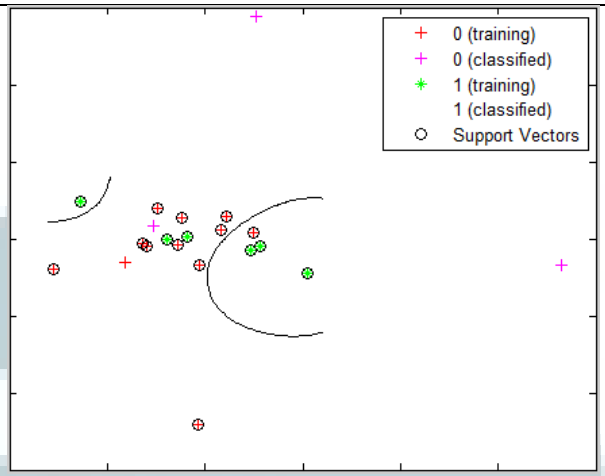
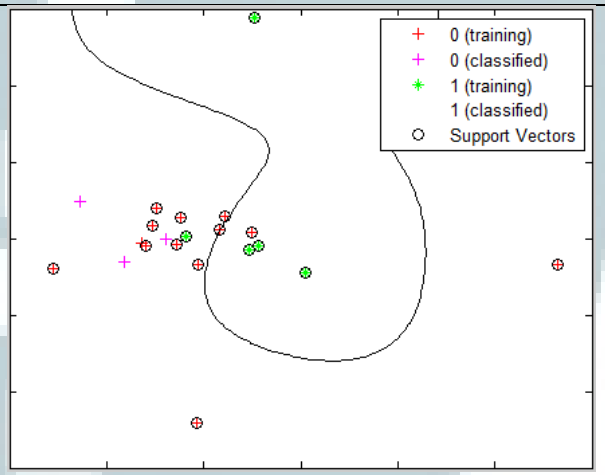
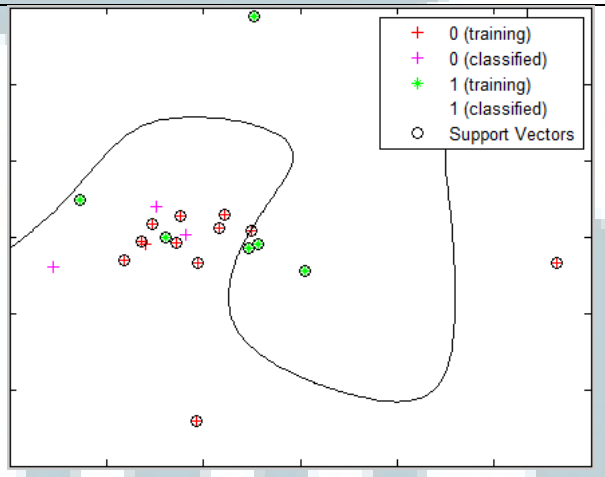
$$Rata - Rata = \frac{66.67 + 66.67 + 66.67 + 66.67 + 33.33 + 100}{6} = 66.68$$

4.3 Representasi hasil klasifikasi

4.3.1 Hasil klasifikasi sampel *testing*

Berikut adalah representasi hasil SVM pada proses testing:

K-Fold	SVM	Akurasi
K-2		66.7%
K-3		66.67%
K-4		66.67%

K-5		66.67%
K-6		33.33%
K-7		100%

Tabel 4.6 Tabel Representasi hasil testing gambar

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian yang dilakukan oleh penulis dapat digunakan dan dikembangkan selanjutnya dalam program atau aplikasi pengklasifikasian gambar batik yang lebih kompleks dan tentunya sebagai pembuktian konsep penerapan Gabor *fiter*, PCA , dan SVM pada *image processing*. Dari penelitian ini, dapat menghasilkan kesimpulan sebagai berikut:

1. Gabor *filter* terbukti dapat mengekstraksi nilai karakteristik dari pola gambar batik, dengan mengatur parameter yang ada pada Gabor *filter* agar mendapatkan hasil yang diinginkan penulis.
2. *Principal Component Analysis* (PCA) dan *Support Vector Machine* (SVM). Sangat berguna dan berpengaruh dalam menyederhanakan data set berukuran besar, juga berguna untuk mengklasifikasikan nilai-nilai karakteristik dari gambar batik dengan melalui proses *training* terlebih dahulu.
3. Hasil akurasi yang didapatkan penulis dalam penelitian ini melalui percobaan dengan proses *training* dan proses *testing* tidak mencapai target seperti yang telah disebutkan pada tujuan penelitian yaitu sebesar 80%. Akurasi yang didapat saat proses *training* yaitu sebesar 100% dan akurasi yang didapat saat proses *testing* yaitu sebesar 66.68%.

5.2 Saran

Hasil penelitian dapat dijadikan sebagai dasar dalam pengembangan program atau aplikasi mengenai *image processing* pada pola gambar batik yang lebih kompleks. Berikut beberapa saran yang dapat diberikan penulis berdasarkan penelitian ini.

1. Pola gambar yang dapat digunakan sebagai bahan *testing* maupun *training* dapat lebih banyak variasinya. Contohnya untuk pengambilan sampel pola batik yang diambil tidak hanya yang berpola burung, tetapi ditambahkan pola-pola lainnya seperti harimau,perahu,ikan,dan lainnya.
2. Jumlah sampel yang digunakan diperbanyak agar mendapatkan data yang akurat.
3. Parameter pada Gabor *filter* dapat lebih ditingkatkan. Seperti memperbanyak orientasi yang mempunyai simbol ‘theta’ pada Gabor *filter*. Untuk memperbaiki akurasi yang didapat
4. Dalam melakukan *image processing* , penulis menyarankan untuk menggunakan gambar sampel yang mempunyai resolusi yang besar, dikarenakan kemungkinan memiliki *noise* yang sedikit pada gambar.