Lab 4
Pengolahan Citra
Image Segmentation, Feature Extraction, and Object Recognition
Jumat, 8 November 2019

A. Image Segmentation

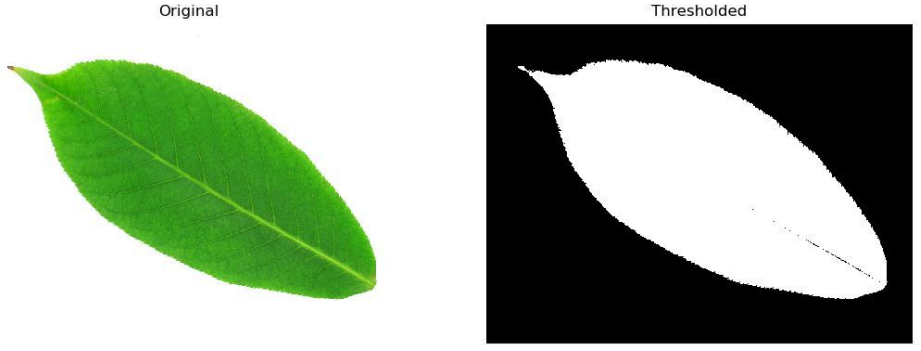
1. Histogram Thresholding (Lib)

Scikit-image memiliki beberapa fungsi untuk threshold pada library `filters`. Berikut ini contoh thresholding menggunakan metode Otsu.

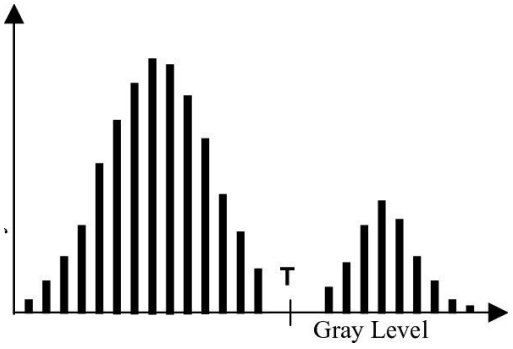
```
from skimage import io, color, filters, util
import matplotlib.pyplot as plt

i1 = io.imread('1101.jpg')
G = util.img_as_ubyte(color.rgb2gray(i1))
T = filters.threshold_otsu(G)
S = util.img_as_float(G > T)

plt.subplot(1,2,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(1-S, cmap='gray')
plt.title("Thresholded"); plt.axis("off")
plt.show()
```



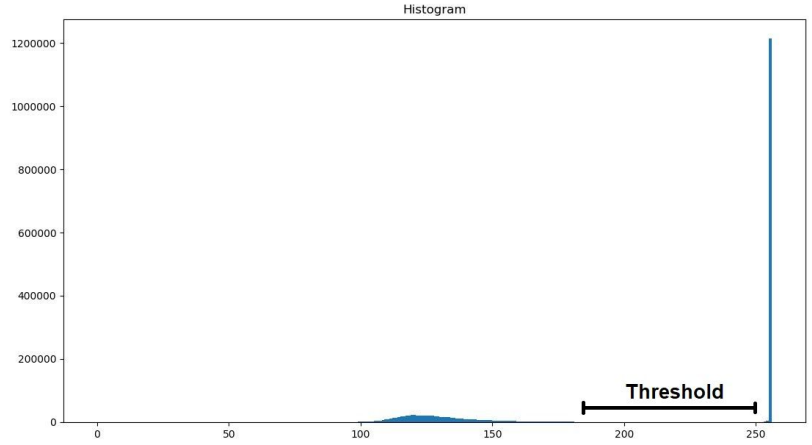
2. Histogram Thresholding (Manual)



Kita dapat memisahkan citra kedalam area ‘terang’ dan 'gelap' dengan menggunakan Thresholding (T).

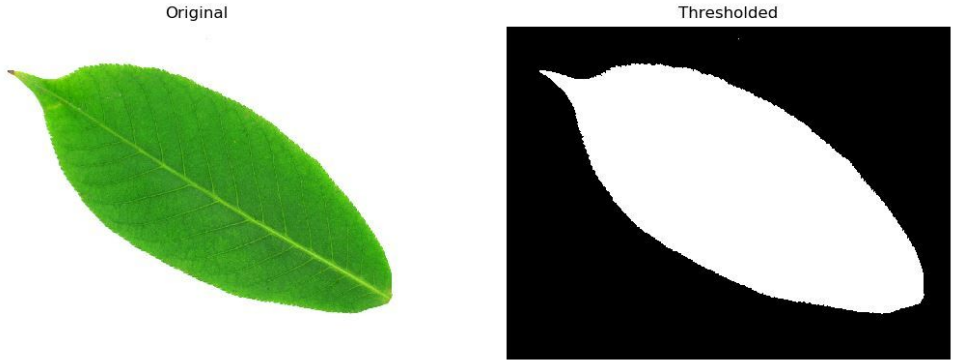
$$g(x,y) = \begin{cases} 1, & \text{if } f(x,y) > T \\ 0, & \text{if } f(x,y) \leq T \end{cases}$$

```
plt.hist(G.flatten(), 256, range=(0,256))
plt.title('Histogram'); plt.show()
```




```
m, n = G.shape
t = 220
for i in range(m):
      for j in range(n):
            if (G[i,j] <= t):
                  G[i,j] = 0
            else:
                  G[i,j] = 1

plt.subplot(1,2,1); plt.imshow(i1)
plt.title('Original'); plt.axis("off")
plt.subplot(1,2,2); plt.imshow(1-G, cmap='gray')
plt.title("Thresholded"); plt.axis("off")
plt.show()
```



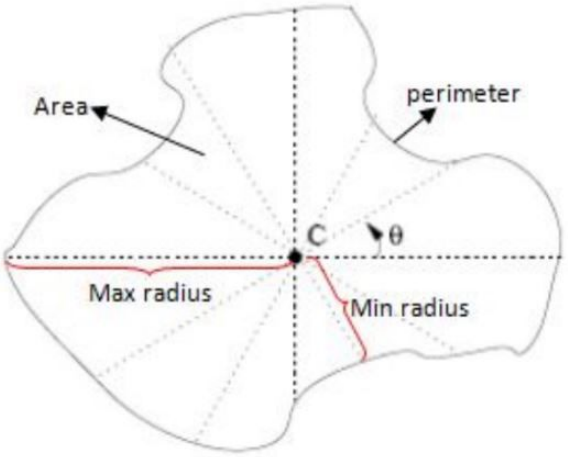
B. Feature Extraction

Ekstraksi fitur merupakan tahapan mengekstrak ciri/informasi (fitur) dari objek di dalam citra yang ingin dikenali/dibedakan dengan objek lainnya. Ciri atau fitur yang telah diekstrak

kemudian digunakan sebagai parameter/nilai masukan untuk membedakan antara objek satu dengan lainnya pada tahapan identifikasi/klasifikasi. Beberapa ciri yang umumnya diekstrak dari suatu citra adalah sebagai berikut:

1. **Ekstraksi Fitur Warna**
   Untuk membedakan suatu objek dengan warna tertentu dapat menggunakan nilai hue yang merupakan representasi dari cahaya tampak (merah, jingga, kuning, hijau, biru, ungu). Nilai hue dapat dikombinasikan dengan nilai saturation dan value yang merupakan tingkat kecerahan suatu warna. Untuk mendapatkan ketiga nilai tersebut, perlu dilakukan konversi ruang warna citra yang semula RGB (Red, Green, Blue) menjadi HSV (Hue, Saturation, Value).

2. **Ekstraksi Fitur Bentuk**
   Bentuk merupakan salah satu fitur yang dapat diperoleh dari sebuah citra. Bentuk adalah informasi geometris yang tetap ketika efek lokasi, skala, dan rotasi dilakukan terhadap sebuah objek (D.G. Kendall).
   Pada scikit-image di Python tersedia fungsi `regionprops` pada library `measure` (https://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.regionprops) yang terdiri dari beberapa fungsi untuk membuat fitur bentuk seperti:
   - Area merupakan luas suatu objek yang dinyatakan dalam jumlah piksel yang terdapat pada objek tersebut.
   - Perimeter atau keliling menyatakan panjang tepi suatu objek.
   - Radius minimal / Minor_Axis_Length merupakan jarak terpendek antara pusat massa dan titik dalam kontur.
   - Radius maksimal / Major_Axis_Length merupakan jarak terpanjang antara pusat massa dan titik dalam kontur.
   - Eccentricity merupakan nilai perbandingan antara jarak *foci ellips minor* dengan *foci ellips mayor* suatu objek.



   Perlu diingat bahwa, untuk membuat fitur bentuk dari sebuah citra, citra masukan harus berukuran 2D sehingga untuk citra 3D atau citra berwarna harus dikonversi terlebih dahulu menjadi citra grayscale atau citra biner.

3. **Ekstraksi Fitur Tekstur**

   Untuk membedakan tekstur objek satu dengan objek lainnya dapat menggunakan ciri statistik orde pertama atau ciri statistik orde dua. Ciri orde pertama didasarkan pada karakteristik histogram citra. Ciri orde pertama umumnya digunakan untuk membedakan tekstur makrostruktur (perulangan pola lokal secara periodik). Ciri orde pertama antara lain: mean, variance, skewness, kurtosis, dan entropy. Sedangkan ciri orde dua didasarkan pada probabilitas hubungan ketetanggaan antara dua piksel pada jarak dan orientasi sudut tertentu. Ciri orde dua umumnya digunakan untuk membedakan tekstur mikrostruktur (pola lokal dan perulangan tidak begitu jelas). Ciri orde dua antara lain: Angular Second Moment, Contrast, Correlation, Variance, Inverse Different Moment, dan Entropy.

Berikut diberikan contoh ekstraksi fitur menggunakan fitur bentuk pada citra daun yang telah disegmentasi. Ekstraksi fitur dilakukan dengan menggunakan fungsi regionprops pada library measure dengan mengambil 7 fitur: `equivalent_diameter, perimeter, area, filled_area, convex_area, eccentricity, dan orientation`.

```
from skimage import measure
import numpy as np
import glob

files = glob.glob("daun/train_bw/*.jpg")
train_shape_feature = np.zeros((100,7))
for k in range(len(files)):
      BW = io.imread(files[k])
      sf = measure.regionprops(BW)
      shape_feature = [max(s.equivalent_diameter for s in sf),
      max(s.perimeter for s in sf),
      max(s.area for s in sf),
      max(s.filled_area for s in sf),
      max(s.convex_area for s in sf),
      max(s.eccentricity for s in sf),
      max(s.orientation for s in sf)]
      train_shape_feature[k,:] = shape_feature
np.savetxt('train_shape_feature.csv', train_shape_feature, delimiter=',',
fmt='%f')
```

```
files = glob.glob("daun/test_bw/*.jpg")
test_shape_feature = np.zeros((20,7))
for k in range(len(files)):
      BW = io.imread(files[k])
      sf = measure.regionprops(BW)
      shape_feature = [max(s.equivalent_diameter for s in sf),
      max(s.perimeter for s in sf),
      max(s.area for s in sf),
      max(s.filled_area for s in sf),
      max(s.convex_area for s in sf),
      max(s.eccentricity for s in sf),
      max(s.orientation for s in sf)]
```

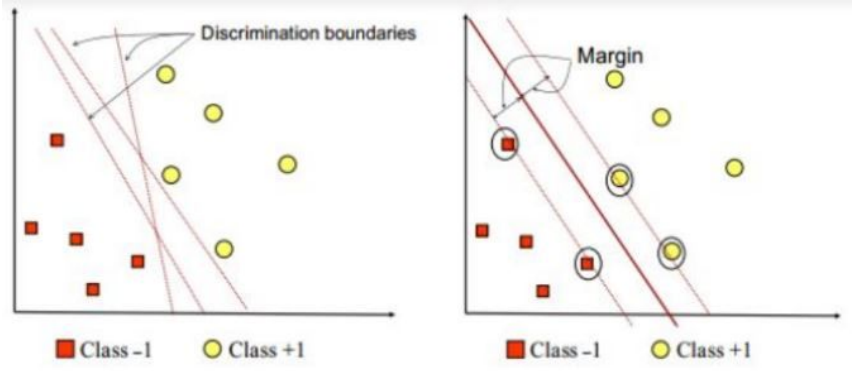
```
      test_shape_feature[k,:] = shape_feature
np.savetxt('test_shape_feature.csv', test_shape_feature, delimiter=',',
fmt='%f')
```

	A	B	C	D	E	F	G
1	1020.589	11787.78	818072	818435	1117059	1	0.785398
2	946.1935	11292.55	703153	703376	986003	1	0.785398
3	1010.252	11808.28	801585	801828	1115790	1	1.306711
4	960.2878	11938.55	724257	724643	956351	1	1.475469
5	1044.573	12218.86	856974	857237	916933	1	1.513223
6	979.7951	12434.58	753981	754359	827767	1	0.785398
7	1036.39	12254.31	843600	843971	902320	1	0.785398

C. Classification

1. SVM

Support Vector Machine (**SVM**) merupakan metode supervised pada machine learning. Proses pembelajaran pada SVM bertujuan untuk memperoleh hipotesis pada hyperplane terbaik yang tidak hanya dapat meminimalkan rata-rata error pada data pelatihan, tetapi juga memiliki generalisasi yang baik. Generalisasi adalah kemampuan suatu hipotesis untuk mengklasifikasi data outlier. SVM mampu bekerja pada data linear dan nonlinear separable. Pada Gambar dibawah ini, terlihat bahwa berbagai alternative hyperplane yang dapat memisahkan semua data ke dalam kelas yang sesuai. Hyperplane terbaik tidak hanya mampu untuk memisahkan data, tetapi juga memiliki margin paling besar.



Alternatif hyperplane (kiri) dan Hyperplane terbaik dengan margin paling besar (kanan)

```
from sklearn import svm, metrics

train_label = np.ones((100,1))
train_label[50:,0] = 2
test_label = np.ones((20,1))
test_label[10:,0] = 2
train_shape_feature=np.genfromtxt('train_shape_feature.csv',delimiter=',')
test_shape_feature=np.genfromtxt('test_shape_feature.csv',delimiter=',')

clf = svm.SVC(gamma='scale')
clf.fit(train_shape_feature, train_label)
svm_label = clf.predict(test_shape_feature)
cm_svm = metrics.confusion_matrix(svm_label, test_label)
```

```
print(cm_svm)
```

```
[[ 6  0]
 [ 4 10]]
```

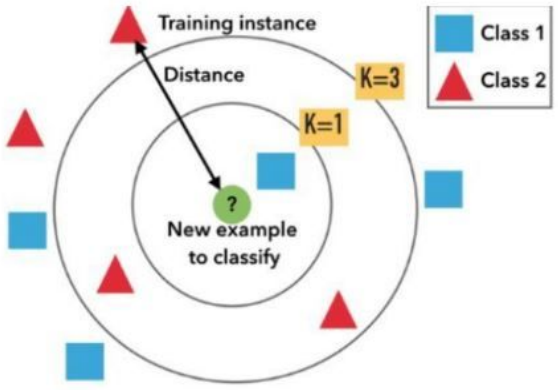
```
def evaluate(actual,predicted):
      """This function evaluates the performance of a classification model
      by calculating the common performance measures: Accuracy,
      Sensitivity, Specificity, Precision, Recall, F-Measure, G-mean.
      Input:
      actual = Column matrix with actual class labels of the training
      examples
      predicted = Column matrix with predicted class labels by the
      classification model
      Output: Row matrix with all the performance measures"""
      p = np.count_nonzero(actual == 1)
      n = np.count_nonzero(actual != 1)
      N = p + n
      tp= sum((int(actual[i] == 1 and predicted[i] == 1) for i in
range(len(actual))))
      tn = sum((int(actual[i] != 1 and predicted[i] != 1) for i in
range(len(actual))))
      fp = n - tn
      fn = p - tp
      tp_rate = tp / p
      tn_rate = tn / n
      accuracy = (tp + tn) / N
      sensitivity = tp_rate
      specificity = tn_rate
      precision = tp / (tp + fp)
      recall = sensitivity
      f_measure = 2 * ((precision * recall) / (precision + recall))
      gmean = np.sqrt(tp_rate * tn_rate)
      return (accuracy,sensitivity,specificity,precision,recall,f_measure)

eval_svm = evaluate(test_label,svm_label)
accuracy = eval_svm[0]*100
sensitivity = eval_svm[1]*100
specificity = eval_svm[2]*100
precision = eval_svm[3]*100
recall = eval_svm[4]*100
f_measure = eval_svm[5]*100
print("Akurasi: %f" % (accuracy))
print("Sensitivity: %f" % (sensitivity))
print("Specificity: %f" % (specificity))
print("Precision: %f" % (precision))
print("Recall: %f" % (recall))
print("f_measure: %f" % (f_measure))
```

```
Akurasi: 80.000000
Sensitivity: 60.000000
Specificity: 100.000000
Precision: 100.000000
Recall: 60.000000
f_measure: 75.000000
```

2. **kNN**

k-nearest neighbor (k-NN atau kNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Proses klasifikasi dilakukan dengan mencari titik c terdekat dari c-baru (nearest neighbor). Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak euclidean.



Ilustrasi kNN

```
from sklearn import neighbors

clf = neighbors.KNeighborsClassifier(3)
clf.fit(train_shape_feature, train_label)
svm_label = clf.predict(test_shape_feature)
cm_svm = metrics.confusion_matrix(svm_label, test_label)
print(cm_svm)
```

```
[[8 1]
 [2 9]]
```

```
eval_svm = evaluate(test_label,svm_label)
accuracy = eval_svm[0]*100
sensitivity = eval_svm[1]*100
specificity = eval_svm[2]*100
precision = eval_svm[3]*100
recall = eval_svm[4]*100
f_measure = eval_svm[5]*100
print("Akurasi: %f" % (accuracy))
print("Sensitivity: %f" % (sensitivity))
print("Specificity: %f" % (specificity))
print("Precision: %f" % (precision))
print("Recall: %f" % (recall))
print("f_measure: %f" % (f_measure))
```

```
Akurasi: 85.000000
Sensitivity: 80.000000
Specificity: 90.000000
Precision: 88.888889
Recall: 80.000000
f_measure: 84.210526
```

```
clf = neighbors.KNeighborsClassifier(7)
clf.fit(train_shape_feature, train_label)
svm_label = clf.predict(test_shape_feature)
cm_svm = metrics.confusion_matrix(svm_label, test_label)
```

```
print(cm_svm)
```

```
[[6 1]
 [4 9]]
```

```
eval_svm = evaluate(test_label,svm_label)
accuracy = eval_svm[0]*100
sensitivity = eval_svm[1]*100
specificity = eval_svm[2]*100
precision = eval_svm[3]*100
recall = eval_svm[4]*100
f_measure = eval_svm[5]*100
print("Akurasi: %f" % (accuracy))
print("Sensitivity: %f" % (sensitivity))
print("Specificity: %f" % (specificity))
print("Precision: %f" % (precision))
print("Recall: %f" % (recall))
print("f_measure: %f" % (f_measure))
```

```
Akurasi: 75.000000
Sensitivity: 60.000000
Specificity: 90.000000
Precision: 85.714286
Recall: 60.000000
f_measure: 70.588235
```