Spencer Bellerose

160795820

bell5820@mylaurier.ca

Assignment 3 - CP467

Directional Histograms

This method is commonly used for things such as pattern recognition. The algorithm goes as follows:
1. thin the image - bring it down to 1 pixel thickness

2. Calculate the image area (total black pixels in thinned image)

3. Make an approximation of the direction of each pixel by fitting the best line passing through the pixel. Therefore, the slops of the line is the direction of the pixel

4. Calculate the number of pixels that have different slopes

5. Normalize each value by dividing it by the area (which is the total number of black pixels)

Our symbol is converted into a vector of length 16 to represent the symbol

Ratios of Histograms

In this method, the features used are histogram distributions. The symbol is divided into 256(16x16) different zones. Next we calculate our histogram distributions. To finish it off, we calculate the ration between the two distributions using the following rule: "The value in slot [i] is equal to the ratio of the difference between he horizontal and vertical projections [i] to their summation" (features.doc, pg. 3), which in other words means we are calculating the ratio of the difference between our distributions to their summation.

Density

Density as a feature is a ratio of the number of black pixels to the total number of pixels in the minimum rectangle that contains the symbol. This is usually used in combination with other features in order to do pattern recognition.

Centre of Mass

Centre of Mass in image processing is a useful feature. When finding centre of mass, we consider the origin of the image to be pixel (0,0), because of this we will denote our center of mass for any object as (Cr, Cc)

Cr and Cc can be calculated using the following calculations:

Cr = 1/n(sigma)x for an n x n image
Cc = 1/n(sigma)y

The Zoning Method

The Zoning Method is where during our training portion, we extract symbol features, the symbols are determined, and finally the symbol and feature are stored in a database.

The process involves dividing the symbol into 16 regions.

format
[x, x, x, x
x, x, x, x
x, x, x, x
x, x, x, x]

After, we determine the normalized pixel density in each zone. The mean vector is then determined for each of the 10 number classes. The data is also used to determine how many symbols lie within the overlap region.

Example of Centre of Mass Calculation

Imports

In [21]:
```
import cv2
import numpy as np
from scipy import ndimage
from matplotlib import pyplot as plt
```

In [22]:
```
# For sign 0
im = cv2.imread('0.png')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 0: ")
print(cen_mass1)
```

```
Center of mass 0:
(130.87579084431667, 98.55744044783572, 0.9998607005767504)
```

In [23]:
```
# For sign 1
im = cv2.imread('1.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 1: ")
print(cen_mass1)
```

```
Center of mass 1:
(188.01337472653933, 168.6360741913896, 0.7887424321351013)
```

In [33]:
```
# For sign 2
im = cv2.imread('2.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 2: ")
print(cen_mass1)
```

```
Center of mass 2:
(242.33432047533717, 131.54693553888492, 1.0)
```

In [25]:
```
# For sign 3
im = cv2.imread('3.png')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 3: ")
print(cen_mass1)
```

```
Center of mass 3:
(118.48426030161622, 107.43654780384234, 0.9998840839547863)
```

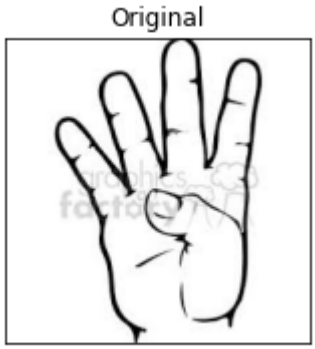
In [26]:
```
# For sign 4
im = cv2.imread('4.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 4: ")
print(cen_mass1)
```

```
Center of mass 4:
(113.27692978968186, 111.55621850053149, 1.0)
```



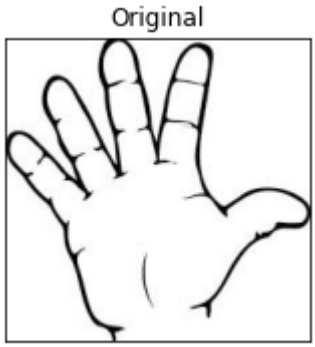
In [27]:
```
# For sign 5
im = cv2.imread('5.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 5: ")
print(cen_mass1)
```

```
Center of mass 5:
(113.09195170609276, 113.19793680635891, 1.0)
```



In [28]:
```
# For sign 6
im = cv2.imread('6.png')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 6: ")
print(cen_mass1)
```

```
Center of mass 6:
(164.19336027036306, 73.86992832207446, 1.0)
```

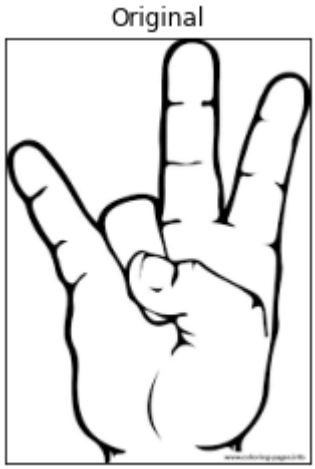
In [29]:
```
# For sign 7
im = cv2.imread('7.png')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 7: ")
print(cen_mass1)
```

```
Center of mass 7:
(133.02318521274248, 93.73149508417096, 1.0)
```



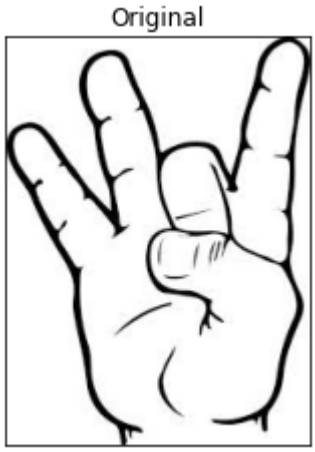
In [30]:
```
# For sign 8
im = cv2.imread('8.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 8: ")
print(cen_mass1)
```

```
Center of mass 8:
(131.0825020811604, 95.78356503124165, 1.0)
```



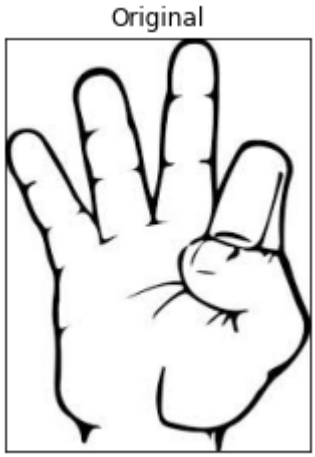
In [31]:
```
# For sign 9
im = cv2.imread('9.jpg')
img = cv2.cvtColor(im, cv2.COLOR_BGR2RGB) #converts BRG to RGB

#show original image
plt.subplot(121), plt.imshow(img)
plt.title('Original'), plt.xticks([]), plt.yticks([])

#Calculate Centre of Mass
lbl = ndimage.label(img)[0]
cen_mass1 = ndimage.measurements.center_of_mass(img, lbl)

print("Center of mass 9: ")
print(cen_mass1)
```

```
Center of mass 9:
(131.56956911360624, 95.39666431628078, 1.0)
```



As we can see from our 10 digits of sign language, we have unique centre of mass for each individual number which can be used as a feature.