cs419 homework3

Doruk Benli

December 2023

Introduction

The aim of this homework is to develop some descriptors to label images with feature extraction. This is done by extracting training feature vectors and for each of the test feature vector, we compare trainings with various distance metrics such as Eucledian, Manhattan(Cityblock), Mahalanobis and chi squared. We determine the performance of descriptors by correct guess of its label/class name. Images classes are directly their names so whenever we predict, we check the image's name to see if it belongs to correct class, if it does we increment the counter by 1, our accuracy is correct predictions divided by test data count.

Implementation Details

The descriptors implemented is described as follows

calculate_area

Calculates the area of the largest contour in the image.

- **Parameters**: Receives a binary image.
- **Returns**: Area of the largest contour.
- **Method**: Uses `cv2.findContours` and `cv2.contourArea`.

calculate_perimeter

Calculates the perimeter of the largest contour in the image.

- **Parameters**: A binary image.
- **Returns**: Perimeter of the largest contour.
- **Method**: Employs `cv2.arcLength`.

calculate_convexity

Computes the convexity ratio of the largest contour.

- **Parameters**: Binary image.
- **Returns**: Convexity ratio, defined as the contour area divided by the convex hull area.
- **Method**: Uses `cv2.convexHull` and `cv2.contourArea`.

calculate_circularity

Determines the circularity of the largest contour.

- **Parameters**: Binary image.
- **Returns**: Circularity, calculated using the formula $(4\pi \times \text{area})/\text{perimeter}^2$.
- **Method**: Utilizes area and perimeter calculations.

calculate_rectangularity

Calculates the rectangularity of the largest contour.

- **Parameters**: Binary image.
- **Returns**: Rectangularity, defined as the contour area divided by the area of the bounding rectangle.
- **Method**: Uses `cv2.boundingRect` and `cv2.contourArea`.

calculate_eccentricity

Measures the eccentricity of the largest labeled region.

- **Parameters**: Binary image.
- **Returns**: Eccentricity of the largest region.
- **Method**: Employs `measure.label` and `measure.regionprops` from `skimage`.

fourier_descriptor

Computes the first few Fourier descriptors of the largest contour.(note that we use absolute value to get rid of the complex values)

- **Parameters**: Binary image, number of coefficients.
- **Returns**: Array of Fourier descriptor magnitudes.
- **Method**: Applies `np.fft.fft` on contour points.

shape_histogram

Generates a histogram of distances from the centroid to the contour points.

- **Parameters**: Binary image, number of bins.
- **Returns**: Normalized histogram array.
- **Method**: Computes centroid and distances, then creates a histogram.

moment_invariants

Calculates Hu's moment invariants.

- **Parameters**: Binary image.
- **Returns**: Array of Hu's moment invariants.
- **Method**: Uses `cv2.moments` and `cv2.HuMoments`.

For the distance functions, we use scipy's distance functions directly.

Accuracy scores of different combinations

Basic Descriptors alone

Descriptor	Euclidean	Manhattan	Chi-Squared	Mahalanobis
Area	0.46	0.49	0.55	0.61
Perimeter	0.31	0.31	0.42	0.60
Convexity	0.31	0.31	0.41	0.60
Circularity	0.31	0.31	0.41	0.63
Rectangularity	0.31	0.31	0.41	0.62
Eccentricity	0.31	0.31	0.41	0.61

Table 1: Accuracies of Base Descriptors alone

In the case of only using the Basic descriptors we manage to get a fairly moderate accuracy scores. Here, the worst distance metric is Euclidean and it appears that the best distance metric is mahalanobis.

Combination of basic Descriptors

Now, we will combine the basic shape descriptors with pairs in order to observe the performance change. Here, by combining different descriptors we were able to increase our performance on both Eucleidan and Manhattan, we also have performance increases with Chi-squared and Mahalanobis, but they do not increase as much as the first two. In these runs we are able to get fair enough scores by mostly combining Area, with other descriptors. If we pick another

Descriptors	Euclidean	Manhattan	Chi-Squared	Mahalanobis
Circularity, Rectangularity	0.31	0.31	0.41	0.68
Area, Eccentricity	0.46	0.49	0.55	0.65
Area, Rectangularity	0.46	0.49	0.55	0.66
Area, Circularity	0.46	0.49	0.55	0.67
Area, Convexity	0.46	0.49	0.55	0.65
Area, Perimeter	0.46	0.49	0.55	0.65

Table 2: Accuracies of Different Combination of pairs

pair like Circularity and Rectangularity, which they both try to extract features in a similar way, we do not get too much of a performance increase, the reason might be due to the fact that both Rectangularity and Circularity try to achieve same thing with different shapes so they are adding less information when we compare it with Area and other descriptors that add perimeter/arclength information Here, we got similar results comparing the one's we pair. We will

Descriptors	Euclidean	Cityblock	Chi-Squared	Mahalanobis
Perimeter, Convexity, Circularity	0.31	0.32	0.42	0.69
Area, Perimeter, Convexity	0.46	0.49	0.55	0.68
Area, perimeter, Circularity	0.46	0.49	0.55	0.68
Area, perimeter, Rectangularity	0.46	0.49	0.55	0.69
Area, perimeter, Eccentricity	0.46	0.49	0.55	0.67

Table 3: Accuracies of Different triple Combinations

combine these basic descriptors with ones like moment invariants, shape histograms and fourier descriptors.

Descriptors	Euclidean	Cityblock	Chi-Squared	Mahalanobis
All descriptors	0.46	0.49	0.55	0.75
Fourier, Histogram, Moments	0.31	0.31	0.41	0.57
Area, Convexity, Fourier, Histogram, Moments	0.46	0.49	0.55	0.65
Fourier	0.31	0.31	0.41	0.46
Histogram	0.50	0.51	0.55	0.44
Moments	0.15	0.16	0.30	0.15

Table 4: Accuracies of Different Combinations of All shape descriptors

Here by looking at the results, if we combine all of the descriptors we get the best accuracy, as Mahalanobis distance jumps to 75 percent accuracy. On the other hand just using Moment invariants and Hu's 8 moments would give a pretty bad predictions. This may be due to impelementation of moments as well

since opencv implements 7 of the Hu's moments, and the 8th one was added by the developer. Its highly likely that my implementation and OpenCV's implementations differ in this case, hence resulting in a bad accuracy. Overall the best Option for my case is to use all the descriptors with mahalanobis distance in order to get an accurate labeling. Another issue was the Fourier descriptor. All other descriptors except fourier descriptor returned a same size feature veectors for different images(for example area, perimeter etc.. returns a single value for an image), One issue was that Fourier descriptor could return different sized feature vectors for different images, to deal with this issue, we iterate through the feauture vectors of fourier descriptors and find the minimum length one and truncate all the others to the minimum sized one. Then we add fourier feature vectors as last vector added to the combination. One improvement i can make is that normalizing the fourier descriptor's values since they are very large compared to other descriptors. Normalizing that may improve the Accuracy since large numbers will result in large differences of squares like Euclidean.

Conclusion

As a conclusion through my implementation of different descriptors, i found combining all of them to be most usefull one. In general i found combining feature vectors to be a usefull method since we allow distance metrics to distinguish more accurately between them. In addition as described above, there had to be some adjustment made for the Moment invariants and Fourier feature vectors in order to make the code run. Since my additional 8th moment is different then opencv's implementations, its causing a bad accuracy result. In addition truncating the fourier feature vectors also causes a loss in performance, but its required to do this properly since we can't measure the distance otherwise. Alternative methods like padding with placeholder values, such as zero, were considered but ultimately discarded due to the introduction of further inaccuracies. In conclusion, while combining multiple descriptors proves to be the most effective approach in my findings, it also necessitates careful consideration of the impact of each descriptor on the overall performance.