

Research on Case-Based Teaching of Digital Image Processing Implemented with Python Language

Taisong Xiong^{1*}, Zhipeng Yang¹, Tao Liu¹, Yifei Zhao¹

¹Chengdu University of Information Technology, Chengdu, 610225, P.R. China

DOI: <https://doi.org/10.36347/sjahss.2024.v12i12.001>

| Received: 06.11.2024 | Accepted: 13.12.2024 | Published: 16.12.2024

*Corresponding author: Taisong Xiong

Chengdu University of Information Technology, Chengdu, 610225, P.R. China

Abstract

Review Article

Digital image processing is a course with strong theoretical backgrounds and also has wide applications in reality. However, traditional teaching method only focuses on delivering abstract theories, which leads to a decrease in students' interest in the course. To resolve this problem, this paper studies the characteristics of basic image processing algorithms from the perspective of combining theory with practice. We investigate the characteristics of fundamental image processing algorithms. We implement some teaching cases using Python.

Keywords: Digital Image Processing, Python Programming, Teaching Methodology, Algorithm Implementation, Practical Application.

Copyright © 2024 The Author(s): This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC BY-NC 4.0) which permits unrestricted use, distribution, and reproduction in any medium for non-commercial use provided the original author and source are credited.

1. INTRODUCTION

Digital image processing is a course that is relatively abstract in theory but also has strong practicality. The course requires students to master basic image processing algorithms and implement them using a programming language. The course of image processing holds an important position in information-related majors. With the rapid development of artificial intelligence technology, the related technologies of image processing are also continuously evolving and updating. Traditional image processing courses teach concepts that are relatively abstract, and the mathematical principles behind the algorithms are also complex for the students. The students often find it difficult to understand and master these digit image processing algorithms. Furthermore, the students don't the applications of the knowledge points they learn, which leads to a decrease in their interest in the course.

Based on the characteristics of the digital image processing course, many teachers construct some relevant cases. They introduce these algorithm principles by combining their implementations. The method induces better effects. However, most image processing cases are implemented based on the Matlab language. However, with the rapid developments of artificial intelligence, Python is an interpretable language. Its code syntax is concise and readable, and it has powerful third-party libraries that enable to implement powerful functionalities. The Python programming language has begun to rise in popularity and has been widely applied in both industry and scientific research fields. According to the latest TIOBE index, Python has consistently ranked first in both 2023 and 2024, and its ratings are more than double that of C++, which is ranked second. The TIOBE index for programming languages is shown in Figure 1.









Nov 2024	Nov 2023	Change	Programming Language	Ratings	Change
1	1		 Python	22.85%	+8.69%
2	3	▲	 C++	10.64%	+0.29%
3	4	▲	 Java	9.60%	+1.26%
4	2	▼	 C	9.01%	-2.76%
5	5		 C#	4.98%	-2.67%
6	6		 JavaScript	3.71%	+0.50%
7	13	▲▲	 Go	2.35%	+1.16%
8	12	▲	 Fortran	1.97%	+0.67%

Figure 1: The TIOBE index for programming languages

We design image processing algorithm cases based on the fact that many algorithms in digital image processing have numerous applications in real life. We explain the principles of the algorithms and demonstrate their effects in the classroom. This method can effectively help students understand the basic principles of image processing algorithms; at the same time, we help students learn about the innovative applications of image processing algorithms. These cases allow students to gain a deeper understanding of the application and implementation of basic algorithm theories, methods, and models in practical scenarios, guiding students to discover, analyze, and solve problems.

These digital image processing cases based on Python programming not only enables students to firmly grasp the basic principles of digital image processing algorithms. Implementing algorithms with Python programming enables students to master the basic syntax and programming skills of Python, enhancing their hands-on abilities. The students obtain a deeper understanding of the application and implementation of algorithm theories, methods, and models in practical scenarios. The method can gradually guide students to discover, analyze, and solve problems, cultivate their practical skills, and broaden their intellectual horizons.

2. Specific Cases

In this section, we list three digital image processing cases implemented using the Python programming language. The three cases are very representative in the course of digital image processing. In the implementation process of the algorithm, we try not to directly call functions provided by the modules. We implement the corresponding algorithms using the Python language. This method can make students more intuitively understand the principles of the algorithms and their programming implementation. The students

can gain a deeper understanding and mastery of the entire algorithm.

2.1 Histogram

The first example is to implement the function of image histogram. The image histogram counts the number of different pixel levels in an image. Image histograms, due to their low computational cost and numerous advantages such as invariance to image translation, rotation, and scaling, are widely applied in various fields of image processing. The hist function in the matplotlib module, the histogram function in numpy, and the calcHist function in opencv all implement the functionality of image histograms. However, merely demonstrating the effect by calling functions provided by modules does not let students to deeply understand the basic principles and specific implementation effects of image histogram algorithms. We implement the histogram function for images using the Python language. The implementation of the entire histogram function is shown in the function histogram_py. The results obtained by calling matplotlib's hist function with the rwidth parameter set to 1, as well as the results of our own implemented histogram_py function, are shown in Figure 2. From Figure 2, it can be seen that the results obtained from both functions are the same, which indicates that the custom function correctly implements the image histogram function. The implementation of this function, not only do students gain a direct understanding of the meaning of image histograms, but they also deepen their understanding and application of various programming concepts in Python, such as function implementation, arrays, loop statements, and module invocation.

```
def histogram_py(imgfile):
    static_num=np.zeros(256,dtype=np.int32)
    imgdata = cv2.imread(imgfile,
    cv2.IMREAD_GRAYSCALE)
```

```
[rows, cols] = imgdata.shape
for r in range(rows):
    for c in range(cols):
```

```
tmpdata=imgdata[r,c]
static_num[tmpdata]=static_num[tmpdata]+1
return static_num
```

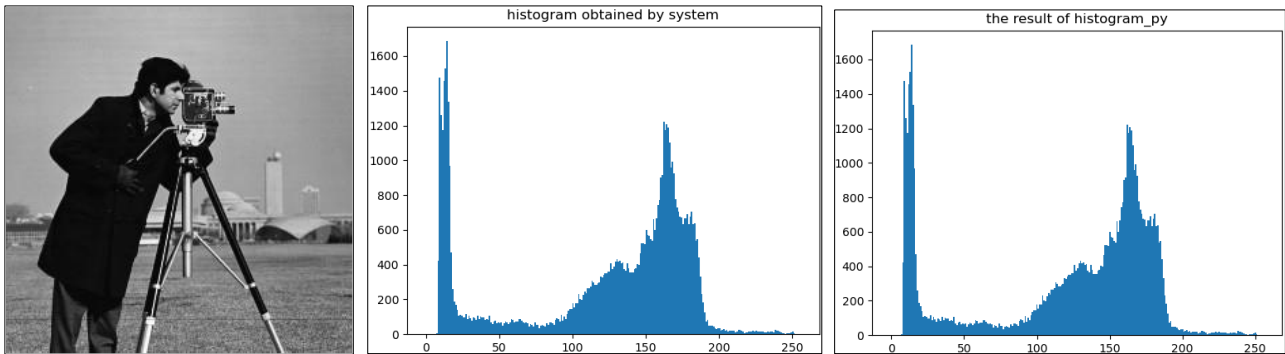


Figure 2: The image and its histogram

2.2 Frequency Domain Filtering

Image Fourier Transform is the process of converting an image from the spatial domain to the frequency domain. Perform filtering operations in the frequency domain, and then convert back to the spatial domain through the inverse Fourier transform. The inverse Fourier transform process is a lossless process. In this case, we implement the two-dimensional discrete Fourier transform of an image based on the Python language, while also performing a centralization operation. The entire image is completed within one period of the Fourier transform. In this case, we perform Gaussian low-pass filtering and Gaussian high-pass filtering operations in the frequency domain, respectively. The function of Gaussian low-pass filtering is to smooth and suppress high-frequency noise, while

Gaussian high-pass filtering is used for sharpening edges and enhancing details. In the experiment, different results were obtained by setting the cutoff frequency values to 10 and 30, respectively. Finally, the results of the filtering operation are subjected to an inverse Fourier transform to convert them back to the spatial domain. The original image, image obtained by Fourier Transform, the results obtained by Gaussian low-pass filtering with $D=10$ and 30 , the results obtained by Gaussian high-pass filtering with $D=10$ and 30 are shown in the Figure 2, respectively. As shown in the Figure 3, the image processed by Gaussian low-pass filter looks clearer with the higher cutoff frequency value. The edge information in the image obtained by the Gaussian high-pass filter with lower cutoff frequency value is more pronounced.

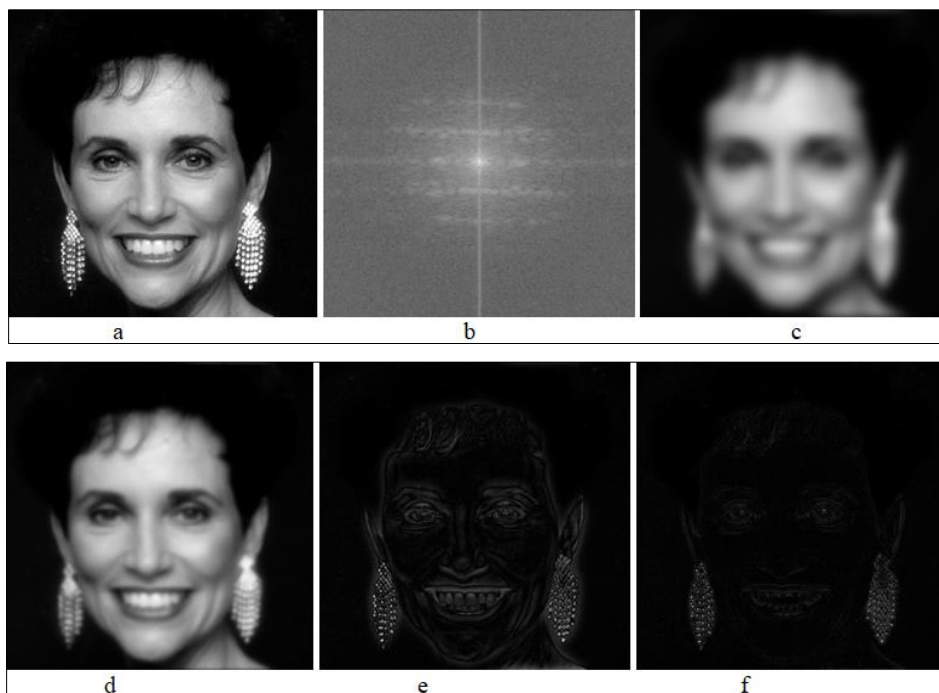


Figure 3: Frequency domain filtering

2.3 Image Segmentation Cases

Image segmentation is an important topic in the field of image processing and has very significant applications in reality. By case-based image segmentation, students can not only gain a good understanding of the basic algorithms of image segmentation but also become proficient in implementing these algorithms. This approach exercises students' hands-on practical skills and enhances their interest in digital image processing. We implement threshold segmentation and K-means segmentation algorithms using the Python programming language. For the two algorithms, we do not directly call the threshold function and k-means function from OpenCV, but instead implement them using the Python programming

language. This approach allows students to gain a more intuitive understanding of the principles behind the algorithms and the specific processes involved in their implementation, thereby deepening their comprehension of the algorithms. Thresholding segmentation algorithms select a threshold based on the overall or partial information of the image and perform segmentation operations based on the grayscale levels. In this case, we provide the original image and its grayscale histogram. Then, we select threshold values of 120 and 200 for segmentation to obtain three targets. The original image, histogram and the segmentation results are shown in Figures 4. The implementation of Thresholding algorithm is given in function `threshold_img`.

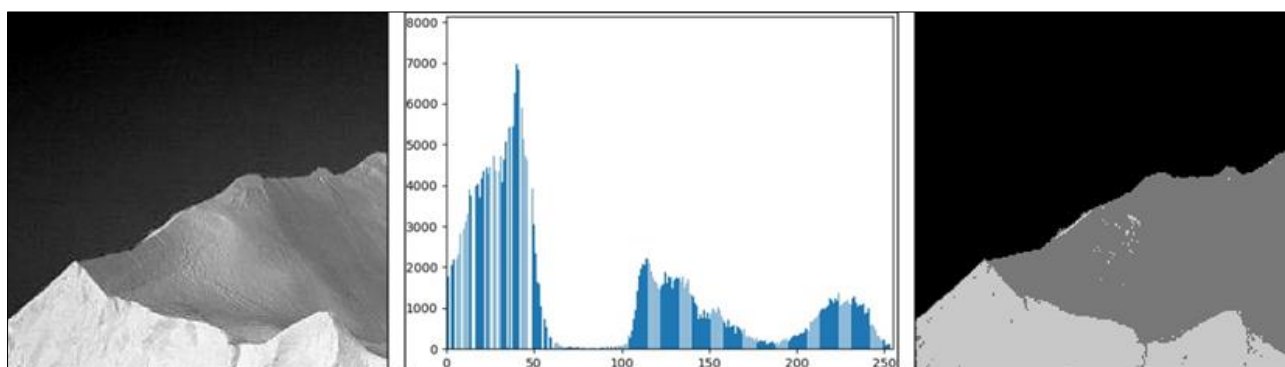


Figure 4: Threshold processing

```
def threshold_img(imgfile):
    imgdata = cv2.imread(imgfile,
cv2.IMREAD_GRAYSCALE)
    [rows, cols] = imgdata.shape
    result_img=np.zeros([rows,cols],dtype=np.uint8)
    for r in range(rows):
    for c in range(cols):
    colorval=imgdata[r,c]
    if colorval<=85:
    result_img[r,c]=0
    elif colorval<=180:
    result_img[r,c]=120
    else:
    result_img[r, c] = 200
    return result_img
```

K-Means is an unsupervised image segmentation algorithm and is also a well-known

clustering algorithm. It mainly uses Euclidean distance as the metric to measure the similarity between data objects. Similarity is inversely proportional to the distance between data objects. The smaller distance represents greater similarity. It generally randomly selects K center points, then compares the distance between image pixels and each center point, then classifies those closest to the same center point into the same category. After the division, the center points of each cluster are recalculated, and this operation is repeated until the change in the center points' distance is less than a certain value. One challenge of the K-Means algorithm is determining the number of center points. The original image and the segmentation results obtained by K-Means with the numbers of center points are 2,3,4 are shown in Figure 5.

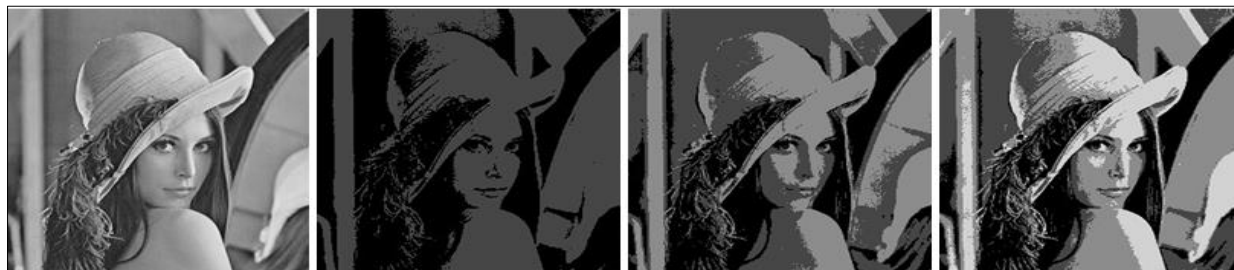


Figure 5: The results obtained by k-means

```

def knn_py(imgfile, knum):
imgdata = cv2.imread(imgfile,
cv2.IMREAD_GRAYSCALE)
[rows, cols] = imgdata.shape
trandata=np.reshape(imgdata,(1,rows*cols))
totalnum=rows*cols
cluster_center =
np.sort(trandata[0,np.random.choice(rows * cols,
knum)])
distance=np.zeros((totalnum,knum),dtype=np.float32)
new_center = np.float32(cluster_center.copy())
absval=100
iternum=0
while absval>0.01 and iternum<30:
iternum=iternum+1
for k in range(knum):
distance[:,k]=np.sqrt((trandata - cluster_center[k])**2)
class_data=np.argmax(distance, axis=1)
cluster_center=np.sort(new_center.copy())
for k in range(knum):
tmpdata=np.mean(trandata[0,class_data==k])
new_center[k]=tmpdata
new_center=np.sort(new_center)
absval=np.sqrt(np.sum((cluster_center-
new_center)**2))/knum
return class_data

```

3. CONCLUSION

We have demonstrated the fundamental principles of digital image processing and the flexible application of various Python syntax and knowledge points by implementing three representative cases of digital image processing, which helps students to master

the course. By implementing the corresponding algorithms, students have deepened their understanding of the principles behind the algorithms and enhanced their practical hands-on abilities. Through these implemented cases, students can become proficient in knowledge such as loops, list and string operations, and the definition and invocation of functions in the Python language. By integrating the knowledge learned, students enhance their ability to analyze and solve practical problems using the acquired knowledge. This lays a solid foundation for students to use the Huawei Cloud platform and CodeArts tools in their subsequent work or research in the field of artificial intelligence.

Funding

This work was supported by Undergraduate Teaching and Educational Research and Reform Project JYJG2023216 and Construction of the Teaching Team for Intelligent Meteorological Detection Technology JYJG2024208.

REFERENCE

- Eric Matthes. Python Crash Course A Hands-on, Project-Based Introduction to Programming. William Pollock. 2016.
- Gonzalez, C., & Richard, E. Woods' Digital Image Processing, Fourth Edition, Global Edition.
- <https://docs.opencv.org/>
- <https://matplotlib.org/>
- Taisong, X., & Yuanyuan, H. (2021). Research on Python Language Teaching Based on Case. *Sch J Arts Humanit Soc Sci*, 9(10), 513-515.