**A Deep Dive into Converting Handwritten Tables in Images to Excel Using Python**

[[Niranj Rajasekaran](https://blog.niranjrajasekaran.com/?source=post_page-----87317f71d31d--------------------------------)](https://blog.niranjrajasekaran.com/?source=post_page-----87317f71d31d--------------------------------)

[[Python in Plain English](https://python.plainenglish.io/?source=post_page-----87317f71d31d--------------------------------)](https://python.plainenglish.io/?source=post_page-----87317f71d31d--------------------------------)

[Niranj Rajasekaran](https://blog.niranjrajasekaran.com/?source=post_page-----87317f71d31d--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F89311ce0489a&operation=register&redirect=https%3A%2F%2Fpython.plainenglish.io%2Fa-deep-dive-into-converting-handwritten-tables-in-images-to-excel-using-python-87317f71d31d&user=Niranj+Rajasekaran&userId=89311ce0489a&source=post_page-89311ce0489a----87317f71d31d---------------------post_header-----------)

Published in

[Python in Plain English](https://python.plainenglish.io/?source=post_page-----87317f71d31d--------------------------------)

·

3 min read

·

Aug 31, 2023

2



In today’s digital age, integrating handwritten content into digital formats is a common requirement. Here, we’ll take a detailed journey into converting handwritten tables from images into Excel using Python, OpenCV, and Tesseract.

**Prerequisites**:  
1. **Python**: Ensure y[ou have Python installed. If not, visit Python’s official website](https://www.python.org/) for installation guidelines.  
2. **Libraries**: Install required Python libraries with:

pip install pytesseract opencv-python openpyxl

1. **Image Pre-processing with OpenCV**:  
Pre-processing is pivotal. It optimizes the image, maximizing the success rate of the OCR.

1.1 **Reading the Image**:

Load the image into Python using OpenCV.

import cv2  
image = cv2.imread('table\_image.jpg')

1.2 **Grayscale Conversion**:  
Color complexities can hinder the OCR process. Converting to grayscale simplifies the image.

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

1.3 **Binarization**:  
Binarization makes the image black and white, further reducing complexities.

\_, binary\_image = cv2.threshold(gray\_image, 128, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

**Note**: `THRESH\_BINARY\_INV` and `THRESH\_OTSU` are thresholding techniques to automate the selection of the threshold value.\*

1.4 **Save for Verification** (Optional):  
It’s often useful to visualize the pre-processed image to ensure accuracy.

cv2.imwrite('processed\_image.jpg', binary\_image)

2. **Text Extraction with Pytesseract**:  
With the image prepared, Tesseract’s power can now be harnessed.

2.1 **Extract Text**:

from pytesseract import image\_to\_string  
extracted\_text = image\_to\_string(binary\_image)

Tesseract examines the image and deciphers the text content, outputting a string.

2.2 **Save for Manual Review**:  
Before moving to Excel, it’s prudent to review the output, ensuring characters have been identified correctly.

with open('output.txt', 'w') as file:  
 file.write(extracted\_text)

3. **Structuring Data for Excel**:  
The extracted text needs formatting before Excel can use it effectively.

3.1 **Setting Up Excel**:

from openpyxl import Workbook  
  
wb = Workbook()  
ws = wb.active

This sets up a new Excel workbook and sheet.

3.2 **Parsing the Extracted Text**:  
The text must be broken down into a tabular format.

rows = extracted\_text.strip().split('\n')  
  
for r\_index, row in enumerate(rows, start=1):  
 columns = row.split('\t')  
  
 for c\_index, cell\_value in enumerate(columns, start=1):  
 ws.cell(row=r\_index, column=c\_index, value=cell\_value.strip())

**Note**: This assumes that rows are separated by new lines (`\n`) and columns by tabs (`\t`). Adjust as necessary based on your data’s format.\*

3.3 **Save to Excel**:

wb.save('output.xlsx')

4. **Manual Refinement**:  
Automated OCR, especially with handwriting, might have inaccuracies. Open `output.xlsx` and verify the content. Look for:

- Misinterpreted characters  
- Merged cells that need splitting  
- Formatting inconsistencies

**Conclusion**:  
By combining Python’s versatility, OpenCV’s image-processing capabilities, and Tesseract’s powerful OCR, the transition from handwritten tables to digital format becomes achievable. Still, due to the variability of handwriting, always anticipate some manual refinement to ensure data integrity.

This guide offers a basic walkthrough, but the realm of image processing and OCR is vast. As you dive deeper, you might encounter more sophisticated challenges, requiring advanced techniques. Yet, with Python at your side, solutions are just a few lines of code away.

**In Plain English**

*Thank you for being a part of our community! Before you go:*

* *Be sure to****clap****and****follow****the writer! 👏*
* *You can find even more content at*[***PlainEnglish.io***](https://plainenglish.io/)***🚀***
* *Sign up for our*[***free weekly newsletter***](http://newsletter.plainenglish.io/)*. 🗞️*
* *Follow us on*[***Twitter***](https://twitter.com/inPlainEngHQ)***(X***), [***LinkedIn***](https://www.linkedin.com/company/inplainenglish/), [***YouTube***](https://www.youtube.com/channel/UCtipWUghju290NWcn8jhyAw), and [***Discord***](https://discord.gg/GtDtUAvyhW)***.***