COMPUTER VISION FOR DATA CATALOGING IN ECOLOGICAL RESEARCH

Shannon Heh

Lynbrook High School, San Jose, California, USA

ABSTRACT

Data collection is an essential, but time-consuming procedure in ecological research. An algorithm was developed by the author which incorporated two important computer vision techniques to automate butterfly cataloguing. Optical Character Recognition is used for character recognition and Contour Detection is used for image-processing. Proper pre-processing is first done on the images to improve accuracy of character recognition and butterfly measurement. Although there are limitations to Tesseract's detection of certain fonts, overall, it can successfully identify words of basic fonts. Contour detection is an advanced technique that can be utilized to measure an image. Multiple mathematical algorithms are used to calculate and determine the precise location of the points on which to draw the body and forewing lines of the butterfly. Overall, 92% accuracy are achieved by the program for the set of butterflies measured.

KEYWORDS

Computer Vision, Image Recognition, Character Recognition, Ecology, Butterfly Cataloguing.

1. INTRODUCTION

Data collection is an important step of scientific research, especially in ecological and evolutionary studies. Scientists must gather a large amount of data to perform analyses of certain species and support their hypotheses. Much of the information is still contained in physical books that include images and descriptions. The traditional way to digitize the data is to manually measure the body sizes of organisms from these images and type the information into documents and spreadsheets. However, this method is time-consuming and inefficient. It requires a lot of manpower and is also prone to inaccurate measurements due to human errors.

During my Earth Science internship at Stanford University, one of my tasks was to go through 50 books, measure the forewing and body length of thousands of butterflies, extract taxonomic information about each species, and record everything in an Excel spreadsheet. The data collection process not only requires knowledge of Lepidoptera classification, but also needs focus and patience. Two interns can only measure 300 butterflies in three hours.

My internship experience inspired me to search for efficient ways to automate the data collection process. While studying computer vision with Professor Susan Fox, I realized the application of computer vision would be the perfect solution to optimize the data collection process for ecological research. The goal of my project is to employ computer vision techniques to facilitate mass data collection and measurement. With computer vision, the ecological data catalogue process can be sped up significantly.

2. BACKGROUND INFORMATION

2.1. What is Tesseract?

Tesseract is an open-source Optical Character Recognition (OCR) engine that can detect approximately 100 languages [1]. In this project, pyTesseract [2], a Python wrapper for Google's Tesseract-OCR Engine, is implemented for character detection. Tesseract follows a step-by-step process for character detection. The first step is a connected component analysis to organize character outlines into "Blobs." Text lines are then found and broken into words based on the spacing between words. Character recognition is described as a "two-pass process": the first pass involves recognizing each word and passing it to an adaptive classifier as training data, while the second pass recognizes words again to ensure that the all words are well-detected [3].

2.2. How are the butterflies measured?

Two measurements are taken to quantify the butterfly's body size. The first is the forewing length (or basal-apical length) which extends approximately from the upper half of the butterfly body to the outer edge of the butterfly's forewing as shown below. The second is the butterfly's body length which starts right below the butterfly's head and ends at the tip of the body. (In my program, the body length will be measured from the *top* of the head to the end of the body.)

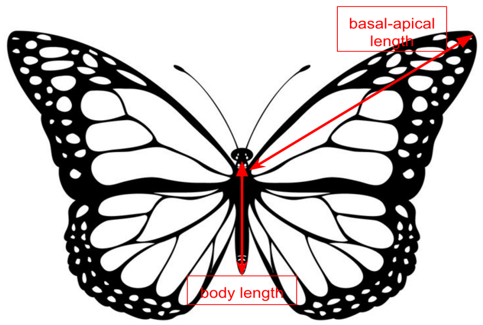
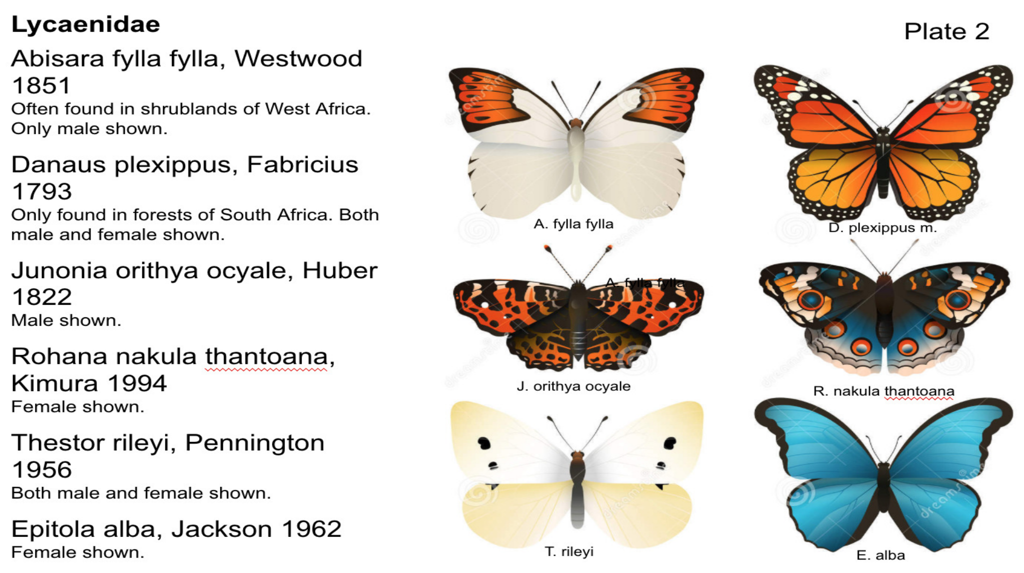


Figure 1. Measurements taken for a butterfly

2.3. How are the butterflies typically displayed in books?

Figure 2 is a mock-up example showing how the butterflies are displayed in books. The data recorded about the butterflies include but are not limited to: the book in which the butterflies are displayed, the butterfly's family, genus, species, subspecies, authority, authority year, sex, page number, figure number, magnitude or scale, forewing length, and body length. The characteristics of the butterfly are often displayed beside or below the butterfly's images. Other descriptions of the butterfly's habitat and behaviours are sometimes included in the books, but unnecessary for data collection about the butterflies' body sizes.



Lycaenidae

Plate 2

Abisara fylla fylla, Westwood 1851
Often found in shrublands of West Africa. Only male shown.

Danaus plexippus, Fabricius 1793
Only found in forests of South Africa. Both male and female shown.

Junonia orithya ocyale, Huber 1822
Male shown.

Rohana nakula thantoana, Kimura 1994
Female shown.

Thestor rileyi, Pennington 1956
Both male and female shown.

Epitola alba, Jackson 1962
Female shown.

Figure 2. A mock-up examples of a typical butterfly display in books

2.4. Related Work

In past studies of computer vision, butterflies were never detected and measured for ecological studies. However, fish have been studied and measured quite often to automate the inefficient manual sorting of fish. D.J. White, C. Svellingen, and N.J.C. Strachan [4] detected the principal axis of the fish by finding the two furthest points on the outline of a fish, an idea I implemented into my own project to determine the butterfly's forewing length.

Various interesting methods were proposed for text detection, including Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Neural Networks, and Tesseract for OpenCV. SVM, KNN, and Neural Networks all implement machine learning techniques to train the system to recognize letters and symbols. SVM [5] is a discriminative classifier defined by a hyperplane that maximizes the margin of training data to find the best match between the characters with which the classifier has been trained and the character needing recognition. Yafang Xue [6] implemented SVM classifiers as one method of OCR, but the algorithm accuracy reached only 83%. P. Kumar, N. Sharma, and A. Rana [7] measured a 94.8% accuracy for the SVM Classifier, but only 80.96% accuracy for Neural Networks. KNN [8] is another classification algorithm which locates the "nearest neighbor," or most similar match, in the training data to the character needing recognition. The data collector can gauge how many, *k*, neighbors should be detected for matching. While SVM and KNN are simple machine learning algorithms for character recognition, the process for training the classifiers is arduous; one needs to train the system for each character and font to ensure the highest accuracy. After reading a study from K.M. Sajjad [9] about Tesseract in automatic license plate recognition, I decided that using Tesseract in my project would be the fastest, most accurate OCR method, since the character recognition is already implemented into the engine.

The papers by Xue, Sajjad, White, and Kumar also presented effective pre-processing techniques, such as binarization, color thresholding, and resizing, that were applied to my butterfly images.

3. Solution

Two main computer vision techniques were investigated for this project
:

1. Optical Character Recognition (OCR): to read the taxonomic information of butterflies
2. Contour Detection: to process the images of butterflies

Optical Character Recognition is the conversion of typed, printed, or handwritten texts into machine-readable texts by a computer. I integrated an existing OCR Engine called Tesseract to conduct the character identification. The butterfly descriptions and measurements are then scanned and input into columns of a spreadsheet using the *openpyxl* library in Python.

Contour Detection was necessary to isolate the shape of the butterfly and locate the points on the butterfly from which to measure their body and wing lengths. Multiple mathematical methods are used to help calculate and detect the locations for measurement. Although I used butterflies as a proxy for measurement, any object can be measured with computer vision.

The solution is divided into two phases: (1) character recognition and (2) butterfly measurement and detection. Phase one incorporates Optical Character Recognition (OCR) to recognize and print the characters in the image, while Phase two manipulates various Python functions and libraries to detect contours, measure linear distances on butterflies, and enter data into a spreadsheet. The two-phrase solution can be applied to scientific research for automated and efficient data collection.

3.1 Character Recognition

Figure 3 is the flow-chart to show how character recognition is implemented.

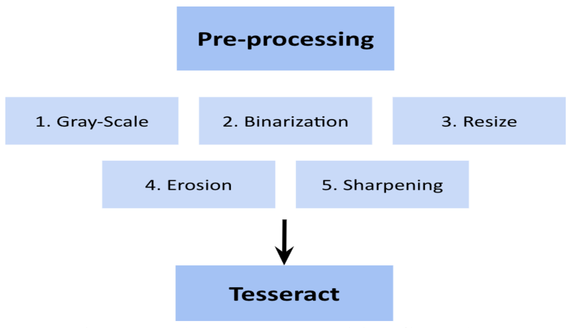


Figure 3. Character Recognition flow-chart

3.1.1 Pre-processing

Pre-processing is necessary to enhance the features of an image. Listed below are the pre-processing techniques that I implemented.

1) Gray-scale: The first step is to convert the image to a gray-scale image which only uses one channel and eliminates extraneous color information. Feature detection is improved, and noise is reduced as a result.
2) Binarization: Binarization converts the image to black and white using a threshold value. Binary thresholding sets values above the threshold to the max value (white), and values less than or equal to the threshold to zero (black).
3) Resize: The next step is to resize the dimensions of the image to two times the original. By testing the code, I found that scaling the text larger makes the character recognition more accurate. The font size is assumed to be around 10-12 points.

4) Erosion: The following pre-processing step is erosion, which adds a layer of pixels to the image and thickens the text.
5) Sharpening: The final pre-processing step is sharpening. Sharpening is used to enhance the edges and features of each character. For example, I found that with one text, the program would confuse the letter '1' and the symbol ']'. After image sharpening, the program was able to make the distinction.

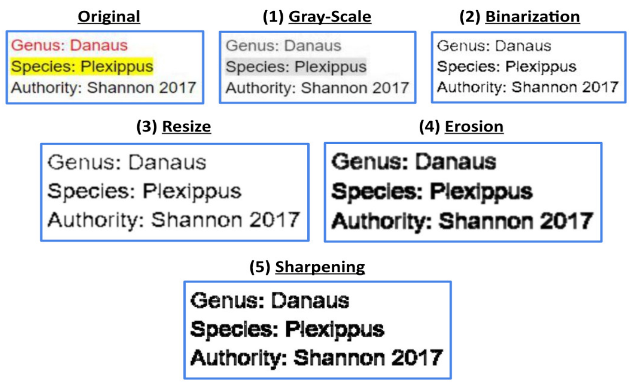


Figure 4. Result of each step of pre-processing

3.1.2. Optical Character Recognition

In the second half of the character recognition process, the program utilizes Tesseract, an open-source Optical Character Recognition (OCR) engine. Developed by HP in 1985, Tesseract is currently one of the most accurate OCR software tools.

3.1.3. Test the Program

I tested the program with three different fonts: sans-Serif (Calibri), Serif (Times New Roman), and Lucida Calligraphy to analyze the different ways the type of font can affect the accuracy. I also included narrow (Arial Narrow), italicized, bold, all capitalized, colored, and highlighted text, the full alphabet, and a text comparison for further analysis.

The standard text I use for the tests is:

Species: Danaus
Genus: Plexippus
Authority: Shannon 2017

Shown in the table (Table 1) below are the results for each font type and characteristic added to the text. Note that kerning and line spacing are detected by Tesseract. All text was written in size 12 font. Serif font was written in Times New Roman and Sans-Serif was written in Calibri.

Table 1. Final results of OCR in the program

Sans-Serif – regular Species: Danaus Genus: Plexippus Authority: Shannon 2017	**Sans-Serif -- all capitalized** SPECIES: DANAUS GENUS: PLB'IPPUS AUTHORITY: SHANNON 2017
Serif – regular Species: Danaus Genus: Plexippus Authority: Shannon 2017	**Serif -- all capitalized** SPECIES: DANAUS GENUS: PLEXEPPUS AUTHORITY: SHANNON 2017
Lucida calligraphy – regular s_pecies: Damn: germs: ?{éxg'gpm Autfiority: Sfianmm 2017	**Lucida calligraphy -- all capitalized** S?ECITS.' DANAHS GEM: Ems AHWORIW: SWN 2017
Arial narrow – regular Species: Danaus Genus: Plexippus Authority: Shannon 2017	**Arial narrow -- all capitalized** SPECIES: DANAUS GENUS: PLEXIPPUS AUTHORITY: SHANNON 2017
Italicized -- Sans-Serif, regular Name: Danaus Plexippus	**Bold -- Sans-Serif, regular** Name: Danaus Plexlppus
Full Alphabet – Serif ABCDEFGHIJKLMNOPQRSTUVWXYZ abodefghijklmnopqrstuvwxyz	**Full Alphabet -- Sans-Serif** ABCDEFGHIJKLMNOPQRSTUVWXYZ ahodefghijklmnopqrstuvwxyz
Red Text -- Sans-Serif Species: Danaus	**Yellow Highlight -- Sans-Serif** Genus: Plexippus
Results on August 19, 2017 Lucida Calligraphy (top), Serif, Sans-Serif, Arial Narrow (bottom) ZN'ame: 'Buttefiffy flutfiority: Sflommm 2017 Species: Sometfiing Name: Butterfly Authority: Shannon 20] 7 Species: Something Name: Butterfly Authority: Shannon 2017 Species: Something Name: Butterfly Authority: Shannon 2017 Species: Something *Note the text is different in this example.	**Results on August 26, 2017** Serif (top), Sans-Serif, Lucida Calligraphy, Arial Narrow (bottom) Name: Butterfly Authority: Shannon 2017 Species: Something Name: Butterfly Authority: Shannon 2017 Species: Something Nam: {Butterfly Autfiority: Sfianmm 2017 s_pecies: Smetfiing Name: Butterfly Authority: Shannon 2017 Species: Something *Note the text is different in this example.

In general, simple fonts (Times New Roman, Calibri, and Arial Narrow) are more accurately recognized by the program. Text written in all capitals produced more incorrect results compared to text written regularly. Sans-Serif fonts, written regularly, were all recognized by Tesseract. When the Sans-Serif text is written in all capital letters, the software produced some errors (e.g. in the word 'Plexippus')." Serif fonts, written both ways, are accurately detected. There is again only one small error in the word "Plexippus," when written in all capitals. Recognition of fonts with ornamentation, such as Lucida Calligraphy, produced the most inaccurate results, with no words being detected correctly. Narrow spacing between letters (Arial Narrow) did not hinder the program's detection; this text was actually detected with the best accuracy.

I input the full alphabet to see whether the program could detect all letters with the two most basic Serif and Sans-Serif fonts. Both fonts' recognition produces the same results, with both c's replaced with o's. I then tested the program with italicized, bolded, colored, and highlighted text. Italicized and bolded text had no effect on the program, beside the small error in "Plexippus" where 'l' was confused with 'i' for the bold text. This may be because the bolding over-thickened the characters (which were already thickened during the eroding during pre-processing). The color and background color of the text did not affect the program's character detection because color information was eliminated in the pre-processing.

I found it interesting that the results of the text recognition changed when recorded on separate days. The earlier results show errors in both Times New Roman and Lucida Calligraphy text. The later results produce errors only in the Lucida Calligraphy text, but the errors are in different characters. The printed text also showed differences in line spacing. Recognition of Times New Roman and Lucida Calligraphy text is improved, while recognition of Calibri and Arial Narrow text are consistently perfect. My hypothesis is that the differences in the image window size affected the clarity of the text. Further research would have to be done to determine the exact causes of the discrepancies.

4.1. Butterfly Recognition and Measurement

Figure 5 shows the flow-chart of butterfly recognition and measurement.

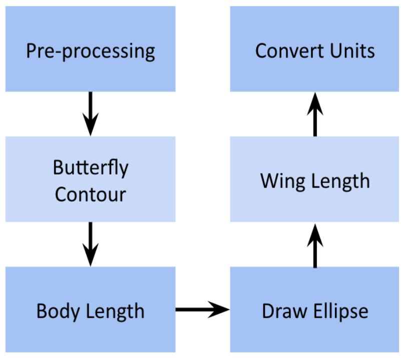


Figure 5. Butterfly measurement flow-chart

4.1.1 Pre-processing

The butterfly image was converted to a gray-scale image to eliminate unnecessary color information. Again, color information makes the image contours more difficult to detect. The image is also sharpened to make contours clearer.

4.1.2 Contour Detection

All the contours on the butterfly image are detected. The outer shape of the butterfly is needed, so only the second largest contour (by area) is chosen. This step is shown in Figure 6.

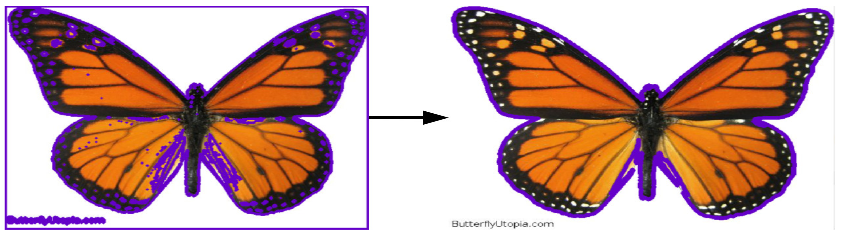


Figure 6. Before and after the largest contour was found

4.1.3 Measuring the Body Length

The following step is to calculate the body length. My initial approach was to locate the midline of the bounding box of the butterfly and find the highest and lowest points within 10 pixels of the midline, which correspond to the head and bottom end of the body. However, this method is ineffective for non-symmetrical butterflies, where the midline is skewed.

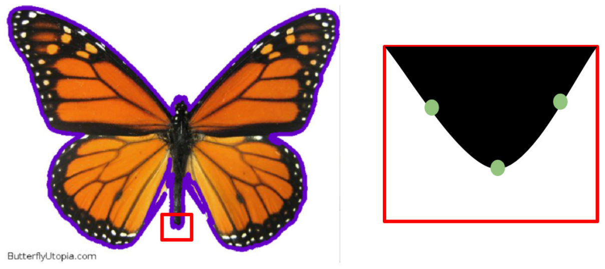


Figure 7. Close-up of the bottom end of the body

My current technique is to locate the bottom end of the butterfly body and calculate its distance from the head. I limited the range of contours to only look at the those within 50 pixels, left and right, of the midline to account for the potential skewness caused by non-symmetrical cases. Next, I chose three points that are 0, 2, and 4 pixels apart from the first. I then tested whether the middle point is the local minima as displayed in Figure 8. A line is directly drawn from the bottom end to the point where the line and head intersect, and the body length is determined.

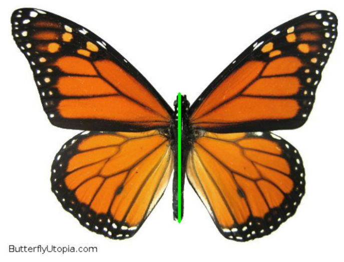


Figure 8. Body length of the butterfly (in green)

4.1.4 Measuring the forewing length

1) Drawing an Ellipse

In order to measure the butterfly's forewing length, I first drew an ellipse, which is similar to the butterfly body shape, around the butterfly's body. The major axis is half of the body length (Figure 9) and the minor axis is around 10 pixels, an approximation of the maximum body width of the butterfly. The minor axis is kept constant because there is usually little variation in the body width of butterflies among different species.

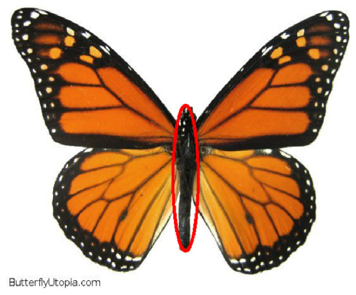


Figure 9. Ellipse around the butterfly's body (in red)

2) Locating Point A

The next task is to find the two points on which to draw the line for the forewing length. Figure 11 shows where the points are usually found. I will call one point A, and the other point B.

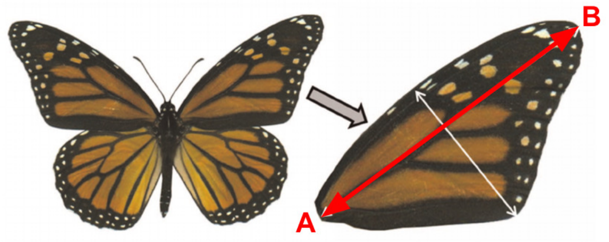
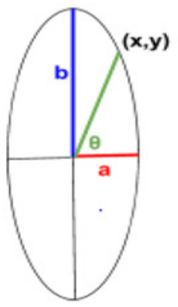


Figure 10. How the butterfly's forewing length is measured

I tested two different methods for detecting Point A. I first tried detecting the top of the butterfly head, then shifted this point 10 pixels down and five pixels to the right, and used that point as Point A. However, because of variations in the butterfly's body size, this method proved unreliable.

After drawing the ellipse, I discovered that I could use the equation of the ellipse to find Point A(x, y) on the body.



$$x = \frac{ab}{\sqrt{a^2 tan^2\theta + b^2}} \quad y = x \cdot tan\theta \qquad (1)$$

After testing multiple angles for θ, I observed that θ should be around π/20.

3) Locating Point B

The last step for the wing length measurement is finding Point B. I predicted that Point B would be the furthest point on the butterfly contour from Point A. Point B, in this case, must be on the right wing, so I made sure that the point was above and to the right of Point A. Finally, the distance formula was used to calculate the distance between Point A and B.

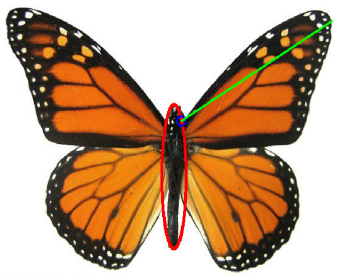


Figure 11. Forewing length of the butterfly (in green)

4.5. Converting from Pixels to Millimeters

The final task for butterfly measurements is converting the body and wing length units from pixels to millimeters. I measured both lengths on a printed image of a butterfly, and used the pixel to millimeter ratio as the scale.

4.6. Separating the butterfly image from the text

The contours around the letters can be a distraction to the program's line and contour detection on the butterfly. To avoid this issue, I separated the butterfly and text into two sub-images. Anything below the lowest point on the butterfly contour would be considered as text (assuming the text is below the butterfly). The character recognition was done on the text image, while butterfly measurements were performed on the butterfly images.

4.7. Sorting the information into an Excel Spreadsheet

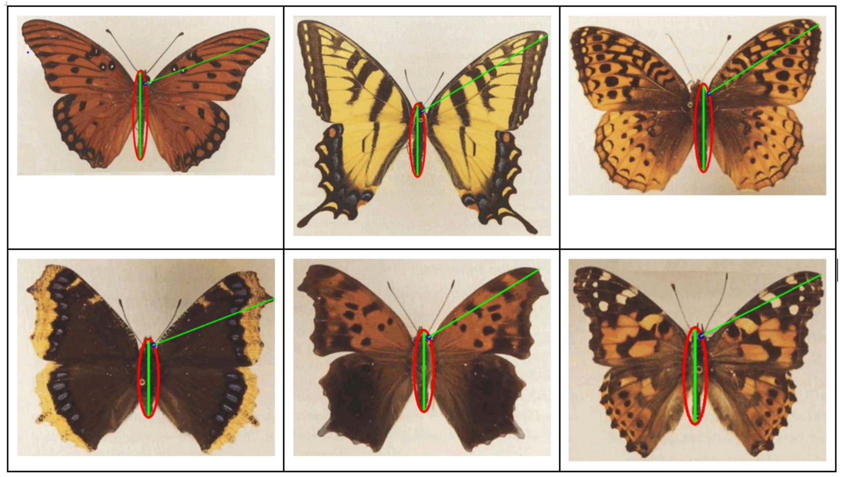
I incorporated the *openpyxl* library in Python to manipulate Excel spreadsheets for data collection. Each time a set of images is passed through the program, a new workbook is generated. The first row of the spreadsheet are the designated headers, as shown in Figure 12. Labels in original image are used for each line of text to identify the genus, species, sex, etc. (in reality, there are no labels for the butterfly's description). Each line of text is then separated into a list of strings. For each butterfly image, a row is appended to the worksheet, and the corresponding information is filled in the cells by matching the text labels to the column headers. An example is shown in Figure 12.

The text labels are detected through fuzzy matching, with the ***SequenceMatcher*** function in the *difflib* library. In case Tesseract incorrectly recognizes some character(s) in the text label, I set the standard so that as long as 75% of the characters in the label are correct, the program can match the label with the header.

Family → YES (100% match) **Famly** → YES (83% match) **Femllv** → NO (50% match)

4.8. Test the Program

The Python program was tested on 12 images of butterflies taken from the book *Learning About Butterflies* by Carolyn Klass and Robert Dirig [10] and various websites. The program goes through all given directories and subdirectories to select the butterfly images.



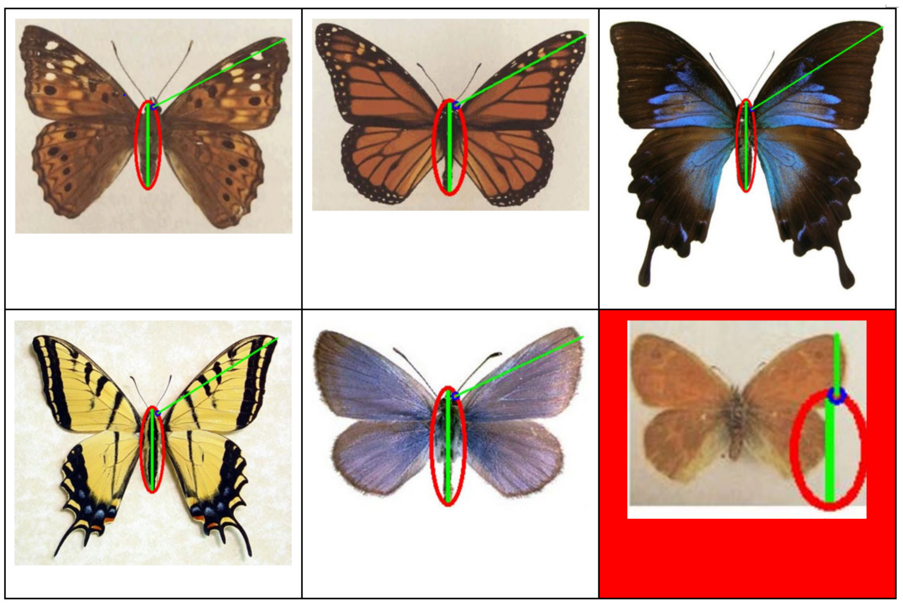
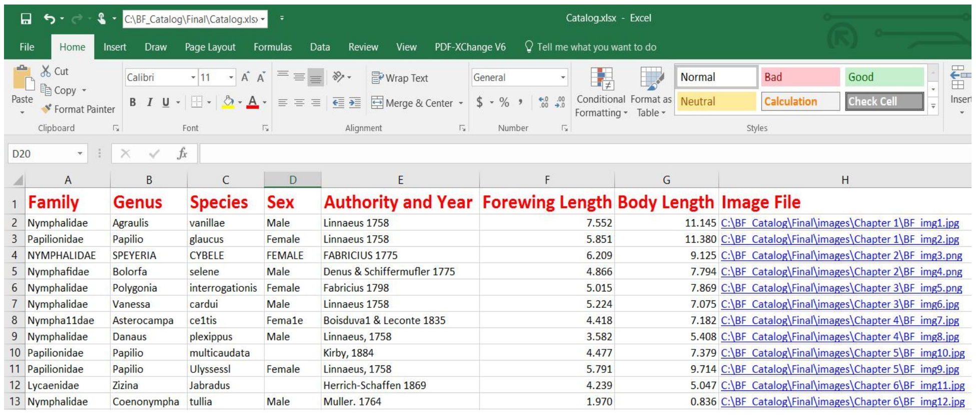


Figure 12. Final result for the 12 butterfly images

Results show that 11 out of 12 (92%) butterflies were successful in being measured and cataloged. The reason that the one butterfly (red box in Figure 14) was not identified correctly is that the butterfly's wing color is similar to the background color. These factors affected the program's ability to distinguish the contour of the butterfly. I also noticed that the rotation of the butterfly image disrupts the program's ability to detect the contours. Because of differences in font for the text, there were minor errors in the character recognition. Overall, this program is proven to be highly accurate and efficient for measurements and data cataloging.



	A	B	C	D	E	F	G	H
1	Family	Genus	Species	Sex	Authority and Year	Forewing Length	Body Length	Image File
2	Nymphalidae	Agraulis	vanillae	Male	Linnaeus 1758	7.552	11.145	C:\BF_Catalog\Final\images\Chapter 1\BF_img1.jpg
3	Papilionidae	Papilio	glaucus	Female	Linnaeus 1758	5.851	11.380	C:\BF_Catalog\Final\images\Chapter 1\BF_img2.jpg
4	NYMPHALIDAE	SPEYERIA	CYBELE	FEMALE	FABRICIUS 1775	6.209	9.125	C:\BF_Catalog\Final\images\Chapter 2\BF_img3.png
5	Nymphafidae	Bolorfa	selene	Male	Denus & Schiffermufler 1775	4.866	7.794	C:\BF_Catalog\Final\images\Chapter 2\BF_img4.png
6	Nymphalidae	Polygonia	interrogationis	Female	Fabricius 1798	5.015	7.869	C:\BF_Catalog\Final\images\Chapter 3\BF_img5.png
7	Nymphalidae	Vanessa	cardui	Male	Linnaeus 1758	5.224	7.075	C:\BF_Catalog\Final\images\Chapter 3\BF_img6.jpg
8	Nympha11dae	Asterocampa	ce1tis	Fema1e	Boisduva1 & Leconte 1835	4.418	7.182	C:\BF_Catalog\Final\images\Chapter 4\BF_img7.jpg
9	Nymphalidae	Danaus	plexippus	Male	Linnaeus, 1758	3.582	5.408	C:\BF_Catalog\Final\images\Chapter 4\BF_img8.jpg
10	Papilionidae	Papilio	multicaudata		Kirby, 1884	4.477	7.379	C:\BF_Catalog\Final\images\Chapter 5\BF_img10.jpg
11	Papilionidae	Papilio	Ulyssessl	Female	Linnaeus, 1758	5.791	9.714	C:\BF_Catalog\Final\images\Chapter 5\BF_img9.jpg
12	Lycaenidae	Zizina	Jabradus		Herrich-Schaffen 1869	4.239	5.047	C:\BF_Catalog\Final\images\Chapter 6\BF_img11.jpg
13	Nymphalidae	Coenonympha	tullia	Male	Muller. 1764	1.970	0.836	C:\BF_Catalog\Final\images\Chapter 6\BF_img12.jpg

Figure 14. Cataloged data in spreadsheet for 12 butterflies

5. Conclusions

Data collection is an essential, but a time consuming and manpower intensive step in ecological research. In this project, two important computer vision methods were implemented to automate data cataloging for ecological study. Optical Character Recognition is an effective approach to scan in written or printed information, which will then be sorted into a spreadsheet. Proper pre-processing is first done on the images before Tesseract can be integrated for character recognition.

Although there are limitations to Tesseract's detection of certain fonts, overall, Tesseract can successfully identify words of basic fonts. Contour detection is an advanced technique that can be utilized to measure an image. Shapes and mathematical calculations are crucial in determining the precise location of the points on which to draw the body and forewing lines of the butterfly. Finally, the butterfly information is input into a spreadsheet and a 92% accuracy are achieved by the program for the set of butterflies measured. While this program is currently limited to butterfly measurements, similar techniques can be applied to the measurement of more organisms.

With the help of computer vision, scientists no longer need to invest significant amounts of their time on data cataloging and measurement. The outcome of this project allows researchers to automate the data collection process and focus more on their research and analyses.

6. Future Work

In this project, I integrated OCR, specifically Tesseract, to automate mass data collection. Tesseract is a highly useful tool for character detection, but there are still some limitations that must be considered. Many factors, such as font size, font type, kerning, and line spacing, must be considered to ensure the most accurate results. I would like to explore more of these limitations, and analyze whether other OCR algorithms, such as K-Nearest Neighbor (KNN) and Support Vector Machine (SVM), can be trained to detect more fonts and handwritten text. Another possible project would be the implementation of a function that recognizes which words are the family, species, genus, sex, and authority and year, without the labels.

Computer-based measurements were also successfully implemented in this project. The next step will be to test the program on more species of butterflies, and to observe whether various wing shapes, patterns, and colors may disrupt the program's detection and yield unexpected results. To achieve a higher accuracy with the 12 butterflies above, I would explore additional pre-processing techniques that could discern the butterfly contour from the background. I would also like to integrate a feature that could rotate a butterfly image so the butterfly's body would be parallel to the vertical edges of the image window. Finally, since the current program is specific to butterflies only, a potential project would be to apply similar techniques in the measurement of other organisms.

Acknowledgements

I would like to thank Professor Susan Fox at Macalester College for mentoring me in computer vision and guiding me through this research project. Thank you to the staff at Pioneer for all your help. Thank you to my Earth Science internship mentor Dr. Noel Heim at Stanford University for your encouragement in pursuing this project.

References

[1] Tesseract OCR (Optical Character Recognition) *Google*. Retrieved September 04, 2017, from https://opensource.google.com/projects/tesseract

[2] pytesseract 0.1.7 *Python Package Index*. Retrieved September 04, 2017, from https://pypi.python.org/pypi/pytesseract

[3] Smith, R. (2007, September). An overview of the Tesseract OCR engine. In Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (Vol. 2, pp. 629-633). IEEE.

[4] White, D. J., Svellingen, C., & Strachan, N. J. C. (2006). Automated Measurement of species and length of fish by computer vision. *Fisheries Research, 80*(2), 3rd ser., pg. 203-210.

[5] Introduction to Support Vector Machines (SVM). Retrieved September 2017 http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[6] Xue, Y. (2014). Optical Character Recognition. *Department of Biomedical Engineering, University of Michigan.* https://pdfs.semanticscholar.org/5b6b/e3357dbdbac38e92515a7b7aebb7e622f635.pdf

[7] Kumar, P., Sharma, N., & Rana A. (2012). Handwritten Character Recognition using Different Kernel based SVM Classifier and MLP Neural Network (COMPARISON) *International Journal of Computer Applications, Volume 53, No. 11, September 2012.*

[8] Understanding k-Nearest Neighbour. Retrieved September 04, 2017, from http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html#knn-understanding

[9] Sajjad, K. M. (2012). Automatic license plate recognition using python and opencv. *Department of Computer Science and Engineering MES College of Engineering*. Retrieved Sept., 2017 http://sajjad.in/content/ALPR_paper.pdf

[10] Dirig, R. & Klass, C. (1992, March) *Learning About Butterflies.* Cornell University, NY: Cornell Cooperative Extension.

[11] Sweigart A. (2015, April 14) Automate the Boring Stuff with Python: Practical Programming for Total Beginners. San Francisco, CA: No Starch Press, Inc.

Author

Shannon Heh is a high school student (11th grade) at Lynbrook High School, San Jose, California, USA. She is President of the Girls Who Code Organization in San Jose for the 2018 to 2019 school year.