Journal of Natural Pesticide Research 10 (2024) 100089

Contents lists available at ScienceDirect

Journal of Natural Pesticide Research

journal homepage: www.journals.elsevier.com/journal-of-natural-pesticide-research

PyPI: An internet-enabled learning tool to boost learner motivation for crop protection

Yoshiyasu Takefuji [*,1]

Faculty of Data Science, Musashino University, 3-3-3 Ariake Koto-ku, Tokyo 135-8181, Japan

ARTICLE INFO

Keywords:
Counting disaggregated objects
PyPI
Software dissemination
Learner's incentive

ABSTRACT

Instructors are always interested in methods to activate learner incentives and motivation to increase learning effectiveness. This paper introduces the Python Package Index (PyPI) as a powerful tool to maximize learner incentives on software and presents an example of its application in agriculture and science. The more useful the PyPI application is, the more it will be downloaded worldwide, providing an external review for the learner, and strengthening their incentive. However, many existing tutorials on PyPI, including the official site, are not updated on the twine library for uploading files to the PyPI site. This paper presents an updated tutorial on using PyPI for counting disaggregated objects such as bugs and pests, and for software reproducibility validation via Code Ocean. Additionally, generative AI is introduced as an indispensable assistant for tasks such as understanding technical terms and providing solutions for encountered problems.

Practitioner Notes

What is already known about this topic

- Instructors don't know how to maximize motivation and incentives of learners.
- Python is widely used in science and technology, including science.
- Instructors have been looking for the best practice to boost the learner's motivation.

What this paper adds

- Internet-enable learning with PyPI can maximize motivation of learners on software.
- The third party evaluation plays a key role in maximizing the learner's motivation.
- The software reproducibility validation is conducted via Code Ocean.
- This may be the first tutorial on software dissemination in pest science and technology.

Implications for practice and/or policy

- Instructors must learn how to debut a PyPI application to disseminate it to the world.
- The more the downloads, the stronger the motivation for learners.
- Disseminating PyPI applications can show skills of learners for possible recruitment.
- Software reproducibility validation via Code Ocean is introduced for scientists.

Introduction

In agriculture, understanding the bug or pest is crucial for crop protection. This includes knowledge of its biology, behavior, ecology, and interactions with other organisms. With this understanding, we can develop effective and sustainable insect management strategies. This paper presents a new application for agriculture, encouraging novice and non-programmers to develop and share their prototype tools with the world. This provides motivation and incentives for the development and dissemination of innovative insect management solutions.

Instructors are always interested in methods on how to improve learner incentives to increase learning effectiveness regardless of research areas. The purpose of this paper is not only to introduce the Python Package Index (PyPI) to maximize learner incentives in higher

* Corresponding author.
E-mail address: takefuji@keio.jp.
[1] **ORCID:** 0000-0002-1826-742X.

https://doi.org/10.1016/j.napere.2024.100089
Received 25 January 2024; Received in revised form 18 April 2024; Accepted 14 July 2024
Available online 16 July 2024
2773-0786/© 2024 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

education and industry education on software, but also to present an example of PyPI applications. The more useful it is, the more its PyPI application will be downloaded worldwide. The more downloads of the PyPI-application, the stronger the incentive for learners on software. PyPI is a powerful internet-enabled tool that helps learners increase their motivation and incentives. It's important for readers to understand that programming languages evolve over time. At present, Python stands as a leading choice for scientific programming.

This paper demonstrates an example of how to debut a Python Package Index (PyPI) application for learners in pest science. The more useful it is, the more its PyPI application will be downloaded worldwide. The proposed example of the PyPI application is to count the number of disaggregated objects such as cells, pests, flies, pollens, dead insect bodies or bug-corpses on a plane, which is widely used in academia and industry. This example is intentionally simple, but powerful for the purposes of this paper for agriculture.

Machine learning has been successfully applied to count the number of objects (Tetila et al., 2020; Falk et al., 2019; Li et al., 2019; Ritch et al., 2020). However, one of the challenges of using machine learning for this task is the need to collect and annotate large datasets, which can be a very time-consuming process. This paper will show the tunable Canny edge detection method with the Gaussian Blur function for noise elimination without machine learning. In other words, the time-consuming annotation is not needed with the proposed method.

According to PePy: https://pepy.tech/, the proposed software, bugcount has been downloaded by 13,915 times worldwide as of April 17, 2024. The large number of downloads of bugcount indicates the popularity. Remember that the more downloads, the stronger the motivation or incentives for learners. The bugcount PyPI is one of the most popular applications of its kind in all areas including pest science.

PyPI has many tutorials on the Internet, but even the official PyPI site has not been updated about uploading files to the PyPI site using the twine library. Following them will fail to upload the PyPI package due to twine's new library changes. To our knowledge, there is no up-to-date tutorial on how to debut a PyPI application.

There are no articles on updated PyPI tutorial on the twine library in the existing tutorials. Thus, the proposed tutorial is significant for advancing science and technology on software.

Learner incentives play an important role in general skill development. Software teachers and instructors are always interested in techniques to activate learner motivation and incentives. The more downloads of PyPI application, the more incentives given to learners. When an outsider in the world who does not know a learner but downloads his/her PyPI application, it is seen as an external review for the learner, so if many downloads occur, it can be judged as an external review very good, thus activating the learner's incentives and motivation.

A numerical blob method was used for a range of diffusive partial differential equations of Wasserstein gradient flow type (Carrillo et al., 2019). Blob stands for binary large object which is a collection of binary data stored as a single entity. The term "large" indicates that only objects of a certain size are of interest and that "small" binary objects are usually noise. Thus, blobs are images in this paper.

The proposed tunable blob method is combined with a Gaussian blur function to eliminate noise for use in real-world applications for counting the number of disaggregated objects. The advantage of the proposed method is its tunable ability to allow users to achieve perfect counting. The disadvantage of the proposed method is that the user needs to do the tuning. This tuning is similar to how many instances of a dataset a user has to prepare in machine learning.

For a large dataset, GPUs are needed in machine learning (Wang et al., 2019; Li et al., 2016; Mittal and Vaishay, 2019). The cost of GPU technology has been dramatically reduced, but GPU computers are still expensive compared to traditional CPU computers. However, applications that can be run on CPU computers without GPU hardware are still highly demanded. The proposed real-time algorithm in this manuscript is based on CPU machines. The second advantage of the proposed method is that it does not require a GPU.

The real-time CPU algorithm means that it runs not only on CPU machines but also on GPU machines.

The conventional counting algorithms using machine learning are based on object detection and object segmentation. Therefore, a large dataset of images for objects detection and objects segmentation is needed in machine learning. Also, collecting and annotating images is a tedious and time-consuming task, while huge datasets of images must be prepared and fed to expensive GPU machines.

In the worst-case, in counting the number of dead insects, a variety of oriented dead images must be used. Besides, deformed corpses (dead bodies) with varying degrees of dryness have too many shapes. In other words, a variety of shapes of objects must be used for correct object detection and segmentation using the conventional machine learning algorithms. In this paper, we do not consider the classification of objects. The conventional machine learning requires the collection of object shapes with various orientations to achieve the goal of accurate counting.

The proposed method is a simplest method without machine learning where the number of disaggregated objects can be counted by using the concept of blob. Blob refers to a group of connected pixels in a binary image (what-when-how.com, 2020; Li and Shui, 2020).

Selecting corpses or a piece of corpses can be determined by the size of object images. The proposed method has a tuning capability with a Canny coefficient. The tunable Canny coefficient allows users to determine the size of object images.

Xiaomin Guo et al. (2013) proposed a cell-counting blob method. However, their program specializes in counting cells and does not use open-source software, so this method cannot be used for other purposes and applications.

Yoshinaga et al. (2010) presented a blob descriptor approach for people counting. However, their approach uses machine learning segmentation. In other words, their method needs tedious machine learning.

Stajnko et al. (2001) proposed a blob method for counting apple fruits. However, their method is based on the colored features such as red color so that their method can only be used for counting colored objects.

The proposed method is based on blobs without color features so that it can be applied to bug-counting, cell-counting, and other counting disaggregated objects.

As far as we know the existing blob methods for counting disaggregated objects, the proposed program, bugcount is one and only one open-source software with the tunable capability which can be installed by de facto standard of PyPI packaging.

In the proposed method, unnecessary noises can be eliminated by repetitive Gaussian Blur functions. Canny edge detection is used for

segmenting target objects. The tunable Canny coefficient in the propose method allows users to determine the size of object images. As far as we know, there is no tunable open-source software for counting disaggregated images which can run on the conventional inexpensive CPU machines.

Canny edge detection and Gaussian blur function are available in OpenCV library of Python based on partial differential equations. The disclosed Python program will run on a notebook PC without a GPU.

The proposed application is one and only one open-source real-time software that runs lightly on a CPU computer. The proposed method has advantages over the existing methods for counting disaggregated objects, which is widely used in industry. However, as mentioned earlier, object classification is not possible with the proposed method.

This paper will show how to debut the packaging software application PyPI to maximize software dissemination in private and business settings. Only a few papers have shown how to debut the PyPI package (Takefuji, 2022a, 2021a, 2021b). PyPI packaging needs three files: README.md, setup.py, and bugcount.py. The PyPI packaging will be detailed in this paper.

In traditional software development, programmers must write programs from scratch. Therefore, a lot of time is required for programming. Rapid open-source prototyping can complete the target program within a few hours. However, selecting the right libraries requires specialized skills.

The contribution of this paper is to select the right library such as OpenCV which is one of the best image processing libraries. However, OpenCV is a complex library so that selecting sub-libraries such as plays a key role in the quality of results. However, in this paper, bugcount allows users to tune two parameters such as Gaussian Blur functions and Canny edge detection by themselves for achieving the satisfiable goal.

In the past, learners had to manually modify Python code and seek help from human experts for unfamiliar tasks in programming and installation. With the advent of generative AI, users can now interact with AI systems to solve problems and manage tasks by themselves.

This tutorial is designed for novice programmers and learners who are new to Python programming. Generative AI systems via web browser such as OpenAI's ChatGPT, Microsoft's Bing.com with ChatGPT-4, and Google's Bard for free of charge can help novice programmers and learners modify code given in this paper for their desired applications. In other words, by providing Python code to a generative AI system, it can automatically convert it to the desired application through interaction with the user, eliminating the need for manual modification. To use a generative AI system, you need to know how to access it, how to provide code via a web browser, and how to use terminal commands to run the modified program. Before running the Python program, you must install Python on your system.

To install Python package, download one of packages depending on your system:

https://docs.conda.io/en/latest/miniconda.html.

It is recommended to download Python 3.9 or 3.8 instead of Python3.10, as generative AI systems may be more familiar with these older versions.

The following access links show three generative AI systems:

ChatGPT-3 via any browser: https://chat.openai.com/.

Bing.com with ChatGPT-4 via Edge browser: https://bing.com/chat.

Bard via any browser: https://bard.google.com/.

If you are not familiar with Python installation, terminal commands, or code editing, you can ask a generative AI system for assistance. It can help with tasks such as running code, installing libraries, and modifying source code. In other words, with the help of generative AI, you no longer need to rely on human experts to solve problems or debug code. The author recommends bing.com with ChatGPT-4 and Bard.

This paper also introduces software reproducibility validation via Code Ocean. The necessary steps are detailed in this paper.

Query to generative AI

The query is crucial in obtaining solutions from generative AI. To achieve their desired goals, users need to possess three skills to effectively navigate generative AI systems:

1. The ability to interact with generative AI using appropriate language and phrasing,
2. the ability to verify the correctness of code generated by generative AI, and
3. The ability to design the code.

The first two skills mentioned in this paper are essential for users. If you encounter errors in the modified code, you can provide them to the generative AI system for correction. You can also ask the AI for details about specific lines of code.

If you are unfamiliar with technical terms or jargon, you can ask a generative AI system for clarification. Current generative AI systems have a limited number of tokens. If you exceed the maximum number of queries allowed via a web browser, the system may stop responding.

One of undergraduate students published a paper on COVID-19 policy outcome analysis tool, jpcovid (Miyagawa and Takefuji, 2023). The jpcovid tool has been downloaded 2796 times which strongly motivated the undergraduate students in programming.

How to install and run bugcount

Before explaining the details of bugcount (Takefuji, 2022b), the following installation procedure shows how to install bugcount via PyPI packaging installation on the system. A PyPI packaging application needs to run Python program. Therefore, you need to install the recommended Python3.8 on your machine. bugcount can run on Windows, MacOS, and Linux operating systems respectively.

1. Download the miniconda from the following site:
   https://docs.conda.io/en/latest/miniconda.html
   You should choose the right file depending on your operating system.
2. Install a Python running environment.
   For Windows, double-click the downloaded file.
   For MacOS, run the following command. ($) sign indicates the prompt from the system in command line terminal.
   $ bash Miniconda3-py38_4.11.0-MacOSX-x86_64.sh
   For WSL on Windows, or Linux operating systems,
   $ bash Miniconda3-py38_4.11.0-Linux-x86_64.sh
3. Run the following command to install PyPI packages such as opencv and bugcount by pip command:

   $ pip install opencv-python

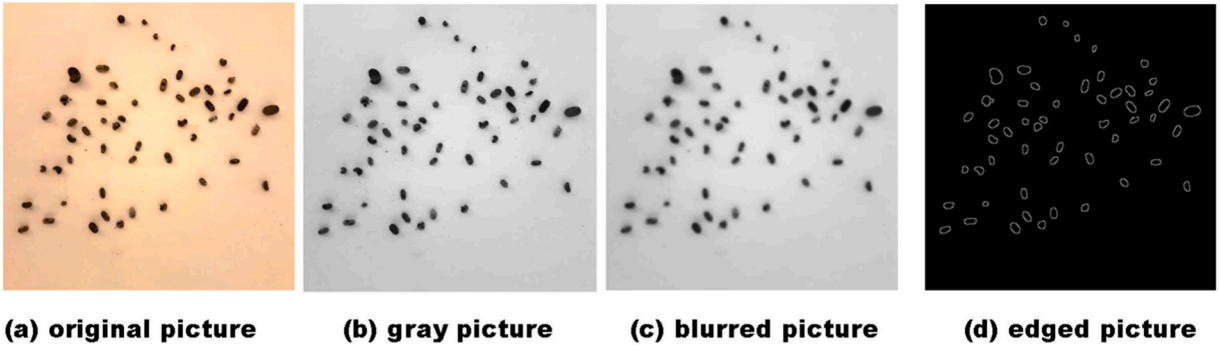

Fig. 1. Four pictures (original, gray, blurred, and edged).

$ pip install bugcount
To run bugcount, you should download a test image:
pillbug.png
Run bugcount by the following command:
$ bugcount pillbug.png
If you will have a trouble to show and pop the result on your screen. You may need to install X-server. For Windows, you should download VcXsrv, Windows X Server exe file and install it: https://sourceforge.net/projects/vcxsrv/.
For MacOS, you should install XQuartz.

Methods

The proposed method is based on the concept of blob. Blob stands for **b**inary **l**arge **ob**ject and refers to a group of connected pixels in a binary image.

The proposed blob method with a tunable capability allows users to achieve perfect counting disaggregated objects without machine learning. The blob method runs on CPU machines that do not have a GPU component. This paper shows a single example of pill bugs with its tuning capability. Supplemental document has two examples.

The proposed method without machine learning is based on tunable Canny edge detection for counting the number of blobs while unnecessary noises can be eliminated by repetitive Gaussian Blur functions. By using Canny edge detection, contours or blobs can be detected and the number of blobs is calculated. In other words, the minimum size of blobs can be determined by a Canny coefficient. By tuning the coefficient, the proposed method can achieve perfect counting of disaggregated objects.

In order to understand the basic algorithm, the program is basically composed of the following six steps:

1. An original RGB color image is read by the Python program.
2. The original RGB color image is converted to grayscale monochrome.
3. The monochrome image is blurred several times by Gaussian Blur function.
4. Canny edge detection is applied to the monochrome image for creating contours (blobs).
5. Counting the number of contours (blobs) formed by detected Canny edges in the monochrome image.
6. Save the result.

Image segmentation using watershed algorithm with Canny edge detection and Gaussian Blur functions plays a key role in segmenting blobs in a given image.

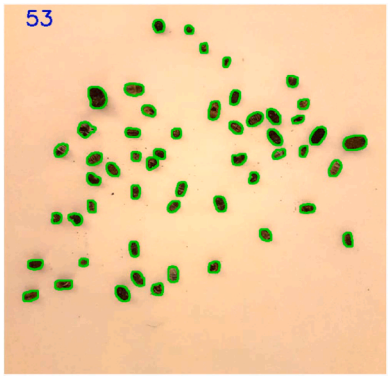

Fig. 2. Generated final result by bugcount (Canny coefficient is 75).

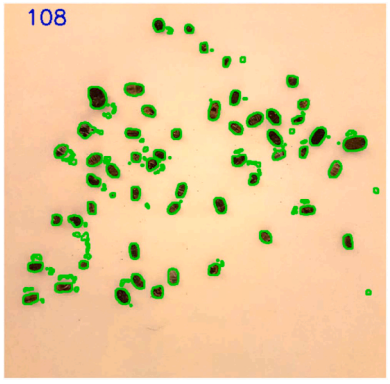

Fig. 3. Generated final result (Canny coefficient is 10).

The problem of Canny edge detection sometimes lies in that any overlapped blobs can be counted as one blob. Therefore, all overlapped-image blobs must be separated or isolated in the target picture on a single plane. In other words, all the objects to be counted need to be disaggregated.

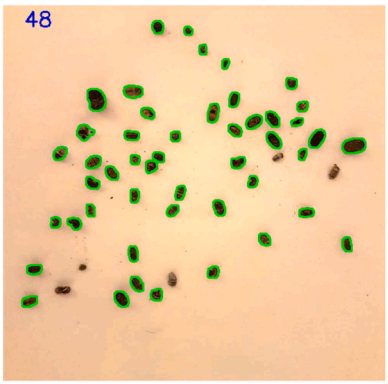

Fig. 4. Generated final result (Canny coefficient is 100).

Results

The proposed method is based on mathematical algorithms using Canny edge detection and Gaussian blur function. This Section empirically explains two functions.

Fig. 1 shows four pictures (original, gray, blurred, and edged) generated by a Python program for counting the number of dead pill bugs. The function of OpenCV Gaussian blur function, cv2.GaussianBlur plays a key role in eliminating cumbersome noises. In the state-of-the-art fine-tuned blob method, the Gaussian Blur function (cv2.GaussianBlur) is called three times in the Python program for eliminating unnecessary noises.

In the fine-tuned program, Gaussian Blur plays a key role in mitigating nuisance noises by blurring an image by Gaussian function.

For novice users running Python programs, a new repository in github site was created for learners:

https://github.com/y-takefuji/counting-for-entomologists.

The new repository shows the followings:

1. how to install an executable Python environment on operating system (Windows, Mac, Linux),
2. how to run bugcount program,
3. how to obtain the final result using three testing pictures (pillbug.png, flies.png, and cells.png),
4. how to resize a picture and how to tune Canny edge detection coefficient.
5. Two other examples including flies counting and cell-counting are tested.

The following command shows and generates the result of five pictures (original, gray, blurred, edged and result).

As mentioned earlier, the proposed method has a tunable Canny coefficient to select blobs of a given image size. This advantageous tunable feature can distinguish between proposed and existing methods. After running the proposed program a few times, the user will be able to recognize the appropriate Canny coefficient that determines the size of a blob.

In this example, an image file of pillbug.png is used for demonstrating the proposed method. Image file and Canny coefficient can be given to bugcount where default Canny coefficient is 75:

$ bugcount pillbug.png.

Fig. 1 shows the result of four pictures (original: orig.png, gray: gray.png, blurred: blur.png, and edged: edge.png). Fig. 1 shows three processes: a process converting a color image into a gray-scaled image, a process blurring a gray-scaled image, and a process of Canny image edging for forming contours (blobs).

Fig. 2 shows the final result picture where 53 dead corpses are counted in a pillbug.png file by running bugcount.

If we set the Canny coefficient to 10, we can detect more blobs, as shown in Fig. 3.

$ bugcount pillbug.png 10.

When the Canny coefficient is set to 100, a small number of blobs are detected, as shown in Fig. 4.

$ bugcount pillbug.png 100.

Software reproducibility validation via Code Ocean

Refereed journals may ask authors to validate the reproducibility of the proposed software as requirement. Code Ocean is one of publishing the reproducibility badge for free of charge. You must create a new account on Code Ocean: https://codeocean.com/.

The following steps are needed for software reproducibility validation:

1. Click New Capsule.
2. Create Blank Capsule.
3. In Environment, click the necessary buttons for installing Python Libraries. In this example, pick Python3.8.5 and pip3. And install matplotlib, opencv-python, and bugcount by add button.
4. Click Start with Sample Files.
5. Fill the form of metadata.
6. Modify run file
7. Click “Submit for publication” and check all items.
8. After receiving acceptance from Code Ocean after few days, you can put the badge on reproducibility qualification in the following form in any markdown document in GitHub site.

Discussion

This paper addresses how to modify the proposed program for accommodating new counting applications.

In open-source software, source codes play a key role in accurately understanding the algorithm. The source codes are a collection of structured mathematical expressions with no ambiguity in any computer languages. Therefore, the minimum important raw Python code is explained in a plain English.

In bugcount.py Python program, outline = cv2.Canny(blurred, 0, 75*coeff) is important for counting blobs in a given image where ‘75’ is the default number to be calibrated for accommodating new counting applications. In Canny edge detection, the lower the number, the more blobs are detected. In other words, the minimum size of blobs can be determined by the Canny coefficient. This means that setting the larger Canny coefficient, a smaller number of blobs will be detected.

Another important parameter is the size of the original image. The input file (pillbug.png) is composed of PNG image data, 550 × 540, 8-bit/color RGB. However, this file was resized from the original file (p.jpg) with 4032 × 3024 by the following resize program:

$ python resize.py p.jpg.

In order to count the number of objects using the proposed blob method without machine learning, Canny edge detection coefficient parameter and the file size must be calibrated for accommodating any counting applications. Remember that aggregated blob images cannot be distinguished in Canny edge detection so that aggregated objects in the original image must be disaggregated.

The proposed Python program will be useful for many counting applications in a variety of applications. Supplemental document contains

two examples including flies.png and cells.png. Two examples can be only used for review, since they are all copyrighted.

PyPI packaging

To debut a PyPI packaging, three files such as README.md, setup.py, and bugcount.py are needed. README.md file can be easily created by GitHub site with creating a new repository with a README file option. To create a new account on GitHub, go to https://github.com/join.

You need to create a new account on pypi site: https://pypi.org/account/register.

Assume you have created a new repository for your project with GitHub and PyPI account.

1. Create a new directory of bugcount
   $ mkdir bugcount
   $ cd bugcount
2. Download README.md file from your GitHub site. This is an example.
   $ wget https://github.com/y-takefuji/counting-for-entomologists/raw/main/README.md
3. Download a setup.py:
   $ wget https://github.com/y-takefuji/counting-for-entomologists/raw/main/setup.py
   Modify nine shaded lines.

```
import setuptools
with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()
setuptools.setup(
    name="bugcount",
    version="0.0.10",
    author="Takefuji",
    author_email="takefuji@keio.jp",
    description="A package for counting BLOB objects",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/y-takefuji/counting-for-learners",
    project_urls={
        "Bug Tracker": "https://github.com/y-takefuji/counting-for-learners",
    },
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    package_dir={"": "src"},
    py_modules=['bugcount'],
    packages=setuptools.find_packages(where="src"),
    python_requires=">=3.7",
    entry_points = {
        'console_scripts': [
            'bugcount = bugcount:main'
        ]
    },
)
```

4. Go to your directory with setup.py and README.md. Run the following commands. $ python setup.py install
   $ python setup.py sdist bdist_wheel
   $ rm dist/*.egg
5. Install twine and create a.pypirc file at your home directory. cat. pypirc shows the content of.pypirc file. password can be obtained from your pypi account settings: Click Two factor authentication (2FA) to generate API token. password starts with pypi-.
   $ cd
   $ cat.pypirc
   [pypi]
   username = _token_
   password = pypi-xxxxxxxx
   $ pip install twine
6. The following command will automatically upload three files.

$ twine upload dist/*.

Conclusion

The proposed program (bugcount) is one and only one open-source software without machine learning for counting aggregated objects. The bugcount application does not need time-consuming annotation tasks unlike existing machine learning methods. The bugcount program is available in public and can be installed by PyPI packaging for experimenting own counting disaggregated objects such as pests. PyPI packing allows bugcount to run on Windows, MacOS, and Linux operating systems with Python installed on your system. The proposed program lightly running on the conventional inexpensive CPU machines has a tunable Canny coefficient so that it can achieve perfect counting of disaggregated objects. The bugcount application has been downloaded 12,576 times worldwide so that its usefulness, practicality and applicability was justified by the large number of downloads. However, as a limitation of the algorithm, the proposed method is not capable of classifying aggregated objects or overlapped objects. As future scope, a powerful disaggregation scheme is needed and should be integrated with the proposed method. This paper introduces generative AI as a tool for solving problems and addressing issues without relying on human experts in programming, installation, and debugging. Generative AI may be an indispensable assistant for many tasks, such as understanding technical terms and providing solutions for encountered problems.

Ethics approval and consent to participate

Not applicable.

Authors' contributions

YT completed this research and wrote the program and this article.

Funding

This research has no fund.

CRediT authorship contribution statement

Yoshiyasu Takefuji: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Conflicts of interest/Competing interests

The author has no conflict of interest.

Consent for publication

Not applicable.

Consent to participate

Not applicable.

Code availability

The code is included in the manuscript.

Availability of data and material

Not applicable.

APPENDIX-1

```
————bugcount.py————————
import cv2,sys
from time import sleep
canny=75
def main(f,canny):
img = cv2.imread(f)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (7,7), 0)
blurred = cv2.GaussianBlur(blurred, (13,13), 0)
#blurred = cv2.GaussianBlur(blurred, (13,13), 0)
cv2.imshow("grey scale", gray)
cv2.imwrite("gray.png", gray)
cv2.imshow("blurred", blurred)
cv2.imwrite("blur.png", blurred)
coeff=int((blurred.max()-blurred.min())/100)
if coeff==1: coeff=1
else: coeff=3
outline = cv2.Canny(blurred, 1, int(canny))
outline= cv2.GaussianBlur(outline, (3,3), 0)
cv2.imshow("The edges", outline)
cv2.imwrite("edges.png", outline)
#(_, cnts, _) = cv2.findContours(outline, cv2.RETR_EXTERNAL, cv2.
CHAIN_APPROX_SIMPLE)
#version4
(cnts, _)=cv2.findContours(outline,cv2.RETR_EXTERNAL,cv2.
CHAIN_APPROX_SIMPLE)
new=[]
th=0
max=0
for i in cnts:
```

```
    if i.shape[0]>max:
      max=i.shape[0]
  for i in cnts:
  if max==i.shape[0]:
    continue
  else:
    th=th+i.shape[0]
  th=int(th/(len(cnts)-1))
  for i in cnts:
    if (i.shape[0]>th-15):
      new.append(i)
  cnts=new
  cv2.drawContours(img, cnts, −1, (0, 255, 0), 2)
  cv2.putText(img,str(len(cnts)),(30,30),cv2.FONT_HER-
SHEY_SIMPLEX,1,(255,0,0),2)
  cv2.imwrite("r.png",img)
  cv2.imshow("Result", img)
  print("%i blobs" % len(cnts))
  sleep(2)
  #cv2.waitKey(0)
  cv2.waitKeyEx(4000)
  if len(sys.argv)==1:
    print('image file is needed!')
    sys.exit()
  if len(sys.argv)==2: f=sys.argv[1]
  if len(sys.argv)==3:
    f=sys.argv[1]
    canny=sys.argv[2]
  main(f,canny)
```

APPENDIX-2

```
————————resize.py————————————
from PIL import Image,ImageEnhance as ie
import sys
img = Image.open(sys.argv[1])
w,h=img.siz
eimg=img.rotate(270,expand=False)
img = img.resize((756,int(756*h/w)), Image.ANTIALIAS)
img=ie.Brightness(img)
eimg=img.enhance(1.9)
eimg=eimg.crop((100,10,650,550))
eimg.save("pillbug.png")
```

References

Carrillo, J.A., Craig, K., Patacchini, F.S., 2019. A blob method for diffusion. Calc. Var. 58, 53. https://doi.org/10.1007/s00526-019-1486-3.

Falk, Thorsten, et al., 2019. U-Net: deep learning for cell counting, detection, and morphometry. Nat. Methods 16. https://doi.org/10.1038/s41592-018-0261-2.

Li, Xiaqing, 2016. Performance analysis of GPU-based convolutional neural networks. In: Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP). IEEE.

Li, W., et al., 2019. Automatic localization and count of agricultural crop pests based on an improved deep learning pipeline. Sci. Rep. 9 (2019), 7024. https://doi.org/10.1038/s41598-019-43171-0.

Li, Ou, Shui, Peng-lang, 2020. Subpixel blob localization and shape estimation by gradient search in parameter space of anisotropic Gaussian kernels. Signal Process. 171, 107495 https://doi.org/10.1016/j.sigpro.2020.107495 (ISSN 0165-1684.

Mittal, Sparsh, Vaishay, Shraiysh, 2019. A survey of techniques for optimizing deep learning on GPUs. J. Syst. Archit. 99, 101635.

Miyagawa, T., Takefuji, Y., 2023. A time-series COVID-19 policy outcome analysis tool to measure human behavior from a herd instinct perspective. Health Technol. https://doi.org/10.1007/s12553-023-00759-x.

Ritch, M.D., et al., 2020. AxoNet: a deep learning-based tool to count retinal ganglion cell axons. Sci. Rep. 10 (2020), 8034. https://doi.org/10.1038/s41598-020-64898-1.

Stajnko, D., et al., 2001. Using image processing and analysis technique for counting apple fruits in the orchard. Hortic. Sci.-UZPI.

Takefuji, Y., 2021b. SCORECOVID: a Python package index for scoring the individual policies against COVID-19. Healthc. Anal. 1, 100005 https://doi.org/10.1016/j.health.2021.100005.

Takefuji, Y., 2021a. Python programming in PyPI for translational medicine. Int. J. Transl. Med. 1 (3), 323–331. https://doi.org/10.3390/ijtm1030019.

Takefuji, Y., 2022a. Deathdaily: a Python package Index for predicting the number of daily COVID-19 deaths. Netw. Model Anal. Health Inf. Bioinform. 11 (1), 14. https://doi.org/10.1007/s13721-022-00359-1 (Epub 2022 Mar 20. PMID: 35342683; PMCID: PMC8934376.

Takefuji, Y., 2022b. bugcount is a PyPI package for counting BLOB objects [Source Code]. ⟨https://doi.org/10.24433/CO.4823110.v1⟩.

Tetila, E.C., et al., 2020. A deep-learning approach for automatic counting of soybean insect pests. IEEE Geosci. Remote Sens. Lett. https://doi.org/10.1109/LGRS.2019.2954735.

Wang, Emma, Yu, Wei, Gu-Yeon, Brooks, David, 2019. Benchmarking TPU, GPU, and CPU platforms for deep learning. arXiv preprint arXiv:1907.10701.

what-when-how.com, 2020. BLOB Analysis (Introduction to Video and Image Processing) Part 1, ⟨http://what-when-how.com/introduction-to-video-and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-1/⟩.

Xiaomin Guo, 2013. A method of automatic cell counting based on microscopic image. In: Proceedings of the 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, vol. 1, pp. 293–6.

Yoshinaga, S., et al., 2010. Real-time people counting using blob descriptor. Procedia Soc. Behav. Sci. 2, 143–152.