memories Documentation

Release 1.0.1-stable

Viraj Thakkar

Dec 02, 2021

GETTING STARTED

Memories is an easy to use package that helps to seperate clustered images from files and helps add metadata to files. The following is a basic description of the package.

CHAPTER

ONE

FIRST STEPS

Let's get you started with the very basics of what all can be done with Memories, installing it and getting to know the software.

1.1 About

Memories is an easy to use package that helps to seperate clustered images from files and helps add metadata, borders and save an image to other formats!

1.1.1 Development

Repository https://github.com/veedata/memories

Issue tracker: https://github.com/veedata/memories/issues

1.1.2 License

1.2 Getting started

1.2.1 Requirements

- Python 3.6+

1.2.2 Installation

Install the Python package:

```
pip install memories
```

1.2.3 How to Use

There are 8 functions at the time being:

- `open_image`: Open an image as a numpy array
- `divided_crop`: Takes 3 inputs, the path to the image, the path where the outful folder should be and the number of images present in the input file. It performs the task of dividing a single image into multiple smaller ones.
- `add_date`: Takes input as the image path and the datetime to be added. it will add date when the image was originally taken.
- `bulk_add_date`: Same as addDate, except it will add date to all images in a folder. The inputs are the folder path and datetime.
- `save_image`: Converts a single image into another format
- `make_page`: Creates a year book like page in HTML
- `make_border`: Creates a border around the image
- `rotate_image`: Returns a rotated image

1.2.4 Example

An example code for each appplication:

```
.. code-block:: python
```

import memories

# Add date to images memories.add_date(“./image-1.jpg”, “27/04/2021 12:00:03”) memories.bulk_add_date(“./”, “27/04/2021 12:00:03”)

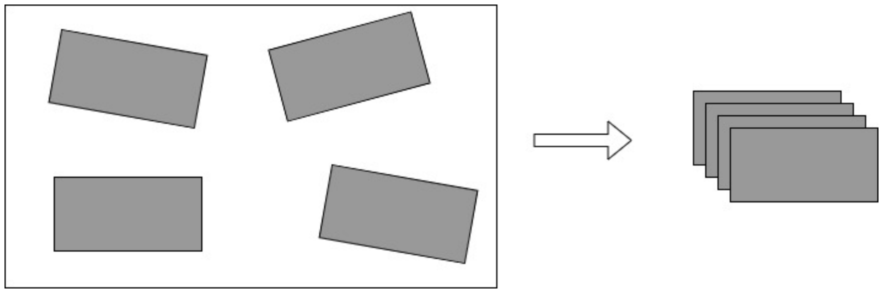
memories.make_page([“./source_folder/image1.png”, “./random/another_source_folder/image2.jpg”], [“CSS”, “Larry”], [“SASS”, “That one got to you, didnt it”], “./save_folder”)

image = memories.open_image(“./image.png”) image1 = mem.open_image(“./image.png”) image2 = mem.open_image(“./image.png”) image3 = mem.open_image(“./image.png”)

memories.divided_crop(image, image_quantity = 6, bgr_value = [255, 255, 255]) # Normal squared borders memories.make_border(image, “normal”, bgr_value = [255, 255, 255], border_dimensions = [100, 100, 100, 100]) # Curved borders memories.make_border(image, “curved”, bgr_value = [255, 255, 255], border_dimensions = [100, 100, 100, 100], radius_dimensions = [100, 100, 100, 100]) memories.save_image(image, “path/to/save_folder/file.extention”) # Save multiple images at once memories.save_image([image1, image2, image3], “path/to/save_folder/file.extention”) # Save multiple images as a pdf memories.save_image([“img-1.png”, “img-1.jpg”, “img-2.jpg”], “path/to/save_folder/file.pdf”)

1.3 Features

1.3.1 Divide Image groups



Converting hard copies of images into their soft copies usually leads to more than a single image being scanned on the same page. And while many devices have an inbuilt option to divide that scan into multiple images, some don’t. This module has been built keeping that scenario in mind. The module divides a scan into it’s member images based on background color also if provided.

1.3.2 Add meta data to images

Allows the addition of metadata to images (only jpg supported). The feature currently only provides addition of Date metadata but will be updated in the future with more options.

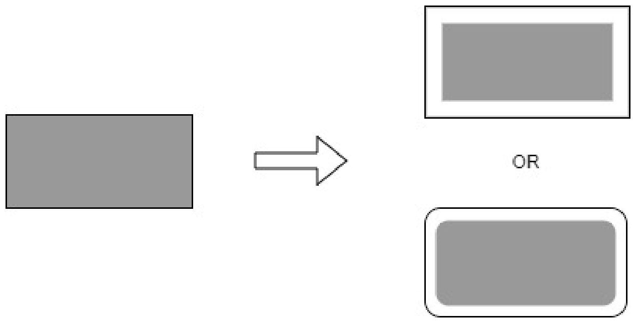
1.3.3 Save in other formats

Easy conversion of an image into other formats. Currently supported input and output formats can be found here. Additionally saving to pdf can be performed, where pdf saves multiple input images in a single pdf file.

1.3.4 Scrapbook

On input of name, short line and image, this function generates a year-book like webpage.

1.3.5 Borders



Make a border around an image. While support is limited over here, it is slated to increase over time, adding new options and developments to the same.

Currently, you can add a normal border to the image and also a curved border. In both options, users can fine tune the width (normal) and radius (curved edges) of the borders.

1.4 Usage

The following section shows a simple approach to be able

1.4.1 Divide Image groups

Divide a image of images into multiple seperate images

```
import memories as mem

image = mem.open_image("./image.png")

mem.divided_crop(image, image_quantity = 6, bgr_value = [255, 255, 255])
```

1.4.2 Add borders to images

Automatically add borders to images. Borders will be added outisde the image and no image content will be cropped.

- borderDimensions = [top-border, bottom-border, left-border, right-border]
- radiusDimensions = [top-left, top-right, bottom-left, bottom-right]

```
import memories as mem

image = mem.open_image("./image.png")

# Squared borders
mem.make_border(image, "normal", bgr_value = [255, 255, 255], border_dimensions = [100,␣
↪100, 100, 100])
# Curved borders
mem.make_border(image, "curved", bgr_value = [255, 255, 255], border_dimensions = [100,␣
↪100, 100, 100], radius_dimensions = [100, 100, 100, 100])
```

1.4.3 Generate HTML page

Allows for generation of HTML page from the input images and text. think of it like an autogenerated scrapbook of sorts.

```
import memories as mem

mem.makePage(["./source_folder/image1.png", "./random/another_source_folder/image2.jpg"],
↪ ["CSS", "Larry"], ["SASS", "That one got to you, didnt it"], "./save_folder")
```

1.4.4 Add metadata

You can even add metadata to images duirectly from the module. (Only works for jpg images currently, future support for png is planned)

```
import memories as mem

# Add date to a single image using image path
mem.addDate("./image-1.jpg", "27/04/2021 12:00:03")
# Add date to images in bulk using folder path
mem.bulkAddDate("./", "27/04/2021 12:00:03")
```

1.4.5 Save file

The save as pdf function takes a list of images as input and produces a pdf with all those images in it, while the normal `saveAs` function is a simple adoption of PIL's save function.

```
import memories as mem

image1 = mem.open_image("./image.png")
image2 = mem.open_image("./image.png")
image3 = mem.open_image("./image.png")

mem.save_image(image1, "path/to/save_folder/file.extention")
# Save multiple images at once
mem.save_image([image1, image2, image3], "path/to/save_folder/file.extention")
# Save multiple images as a pdf
mem.save_image(["img-1.png", "img-1.jpg", "img-2.jpg"], "path/to/save_folder/file.pdf")
```

1.5 Contributing

Thank you for taking the time to contribute! You can contribute in many ways:

1.5.1 Types of Contributions

Suggesting a feature

If you've got a good idea for a feature, then please let us know! Feature suggestions are embraced, but will often be filed for a rainy day. If you require a feature urgently it's best to write it yourself. Don't forget to share ;)

When suggesting a feature, make sure to:

- Check the code on GitHub to make sure it's not already hiding in an unreleased version ;)
- Check existing issues, open and closed, to make sure it hasn't already been suggested

Report Bugs

Report bugs at https://github.com/veedata/memories/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.
- List the steps you've taken so far,
- and any solutions you've tried

1.5.2 Get Started!

Here's how to set up *memories* for local development.

1. Fork the *memories* repo on GitHub.
2. Clone your fork locally:

   $ git clone https://github.com/your_name_here/memories.git

3. Install dependencies:

   $ python -m pip install -r requiements.txt

4. Create a branch for local development:

   $ git checkout -b your-branch-name

   Now you can make your changes locally.

5. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Description of your changes."
$ git push origin your-branch-name
```

6. Submit a pull request through the GitHub website.

1.5.3 Pull Request Guidelines

If you've decided to fix a bug, even something as small as a single-letter typo then great! Anything that improves the code/documentation for all future users is warmly welcomed.

If you decide to work on a requested feature it's best to let us (and everyone else) know what you're working on to avoid any duplication of effort. You can do this by replying to the original Issue for the request.

If you want to contribute an example; go for it! We might not always be able to accept your code, but there's a lot to be learned from trying anyway and if you're new to GitHub we're willing to guide you on that journey.

When contributing a new example or making a change to a library please keep your code style consistent with ours. We try to stick to the pep8 guidelines for Python (https://www.python.org/dev/peps/pep-0008/).

Do

- Do use pep8 style guidelines
- Do comment your code where necessary
- Do submit only a single example/feature per pull-request
- Do include a description of what your example is expected to do
- Do add details of your example to README.md
- The pull request should include tests.
- If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

Don't

- Don't include any license information in your examples- our repositories are MIT licensed
- Don't try to do too much at once- submit one or two examples at a time, and be receptive to feedback
- Don't submit multiple variations of the same example, demonstrate one thing concisely

CHAPTER

TWO

MODULES

Individual modules will be described with examples over here. Needs to be further developed for now.

CHAPTER THREE

LINKS

- genindex
- modindex
- search
- Source code: https://github.com/veedata/memories