L.O.S.E.R.S

Release 1.0

Andrew Seamon

Apr 18, 2021

Table of Contents

Indices and tables

- genindex
- modindex
- search

1.1 classifications.txt

Contains classifications for character recognition training

1.2 constants.py

Colors

constants.**SCALAR_BLACK = (0.0, 0.0, 0.0)**
Built-in immutable sequence.
If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.
If the argument is a tuple, the return value is the same object.

constants.**SCALAR_WHITE = (255.0, 255.0, 255.0)**
Built-in immutable sequence.
If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.
If the argument is a tuple, the return value is the same object.

constants.**SCALAR_YELLOW = (0.0, 255.0, 255.0)**
Built-in immutable sequence.
If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.
If the argument is a tuple, the return value is the same object.

constants.**SCALAR_GREEN = (0.0, 255.0, 0.0)**
Built-in immutable sequence.
If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.
If the argument is a tuple, the return value is the same object.

constants.**SCALAR_RED = (0.0, 0.0, 255.0)**
Built-in immutable sequence.

If no argument is given, the constructor returns an empty tuple. If iterable is specified the tuple is initialized from iterable's items.
If the argument is a tuple, the return value is the same object.

Preprocessing

constants.**GAUSSIAN_SMOOTH_FILTER_SIZE = (5, 5)**
constants for preprocessing.py

constants.**ADAPTIVE_THRESH_BLOCK_SIZE = 19**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

constants.**ADAPTIVE_THRESH_WEIGHT = 9**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

License Plates

constants.**PLATE_WIDTH_PADDING_FACTOR = 1.3**
constants for findPlates.py

constants.**PLATE_HEIGHT_PADDING_FACTOR = 1.5**
Convert a string or number to a floating point number, if possible.

Possible Characters

constants.**MIN_PIXEL_WIDTH = 2**
constants for use with checkIfPossibleChar, this checks one possible char only (does not compare to another char) pixel size

constants.**MIN_PIXEL_HEIGHT = 8**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

constants.**MIN_ASPECT_RATIO = 0.25**
aspect ratio

constants.**MAX_ASPECT_RATIO = 1.0**
Convert a string or number to a floating point number, if possible.

constants.**MIN_PIXEL_AREA = 80**
pixel area

Comparing Characters

constants.**MIN_DIAG_SIZE_MULTIPLE_AWAY = 0.3**
constants for comparing two chars

constants.**MAX_DIAG_SIZE_MULTIPLE_AWAY = 5.0**
Convert a string or number to a floating point number, if possible.

constants.**MAX_CHANGE_IN_AREA = 0.5**
Convert a string or number to a floating point number, if possible.

constants.**MAX_CHANGE_IN_WIDTH = 0.8**
Convert a string or number to a floating point number, if possible.

constants.**MAX_CHANGE_IN_HEIGHT = 0.2**
Convert a string or number to a floating point number, if possible.

constants.**MAX_ANGLE_BETWEEN_CHARS = 12.0**
Convert a string or number to a floating point number, if possible.

Other

constants.**MIN_NUMBER_OF_MATCHING_CHARS = 3**
other constants

constants.**RESIZED_CHAR_IMAGE_WIDTH = 20**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

constants.**RESIZED_CHAR_IMAGE_HEIGHT = 30**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

constants.**MIN_CONTOUR_AREA = 100**
int([x]) -> integer int(x, base=10) -> integer
Convert a number or string to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.
If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

constants.**typeface = 'Typeface.png'**
characterRecognition

constants.**state_names = ['Alaska', 'Alabama', 'Arkansas', 'American Samoa', 'Arizona', 'California', 'Colorado', 'Connecticut', 'District of Columbia', 'Delaware', 'Florida', 'Georgia', 'Guam', 'Hawaii', 'Iowa', 'Idaho', 'Illinois', 'Indiana', 'Kansas', 'Kentucky', 'Louisiana', 'Massachusetts', 'Maryland', 'Maine', 'Michigan', 'Minnesota', 'Missouri', 'Mississippi', 'Montana', 'North Carolina', 'North Dakota', 'Nebraska', 'New Hampshire', 'New Jersey', 'New Mexico', 'Nevada', 'New York', 'Ohio', 'Oklahoma', 'Oregon', 'Pennsylvania', 'Puerto Rico', 'Rhode Island', 'South Carolina', 'South Dakota', 'Tennessee', 'Texas', 'Utah', 'Virginia', 'Virgin Islands', 'Vermont', 'Washington', 'Wisconsin', 'West Virginia', 'Wyoming']**

Used for state recognition in a string

constants.**API_KEY = ''**

assigned at start of LPR_Final.py

1.3 database.py

database.**close** ( )

Closes database connection

database.**connect** ( )

Connects Python to local database with login info from dbConn.py

database.**insert** (*plate, state='', comment=''*)

Used to insert rrecords into database table

database.**select** ( )

Used to select all records in the table. Currently not used.

1.4 dbConn.py

Parameters for use with database.py

host localhost (for now)

user root (not to worry)

passwd n/a (Again. It's a local database.)

database vehicles

table captures

1.5 doubleCheck.py

doubleCheck.**confirmDB** (*origPlate*)

Allows user to add/skip a certain vehicle record. Also allows adding comments.

doubleCheck.**decodeJSON** (*jason*)

Uses regex to split the parsed text from OCR.space. Then using multiple methods, it attempts to differentiate state from plate from [other]. This is done with string length then string matching (check for text in list of states/common issues*) * California and their script typeface:(

doubleCheck.**ensure_dir** (*file_path*)

Makes sure there is a place to output files.

doubleCheck.**getOCR** (*filename*)
Credit: Zaargh | Github OCR.space API request with local file.

Filename Your file path & name.

Overlay Do you need an overlay in the response. Default [F].

Api_key OCR.space API key. Defaults to [helloworld].

Language Language code to be used in OCR. Default [eng].

Returns Result in JSON format.

doubleCheck.**stateExists** (*state*)
Confirms whether a string is a state or not. Uses full name, abbreviation, and common errors.

1.6 FindChars.py

FindChars.**angleBetweenChars** (*firstChar*, *secondChar*)
Use basic trigonometry (SOH CAH TOA) to calculate angle between chars

FindChars.**checkIfPossibleChar** (*possibleChar*)
Confirms whether a specific contour might be a character.

FindChars.**detectCharsInPlates** (*plateList*)
Used to detect any possible characters within the image. Then throws out characters not found inside of a rectangle.

FindChars.**distanceBetweenChars** (*firstChar*, *secondChar*)
Use Pythagorean theorem to calculate distance between two chars

FindChars.**findListOfListsOfMatchingChars** (*listOfPossibleChars*)
Using all possible chars in one list, rearrange the chars into a list of lists containing matching char strings.

FindChars.**findListOfMatchingChars** (*possibleChar*, *listOfChars*)
Given a possible character and a list of possible characters, find all chars in main list that match, and return a list of matches.

FindChars.**findPossibleCharsInPlate** (*imgGrayscale*, *imgThresh*)
Locates all possible characters

FindChars.**loadKNNDataAndTrainKNN** ( )
Loads and runs KNN training to ensure best results

FindChars.**recognizeCharsInPlate** (*imgThresh*, *matchingChars*)
Applies true character recognition

FindChars.**removeInnerOverlappingChars** (*matchingChars*)
If multiple chars overlap, remove the inner char. Prevents including same char twice if multiple contours are found for the same physical char. Ex: 'O' has an outer 'O' and an inner 'o' 'R' has an outer 'R' and an inner 'D'

1.7 FindPlates.py

`FindPlates.`**`detectPlatesInScene`** (*originalImage*)
Finds all possible plates in image by looking for rectangle contours

`FindPlates.`**`extractPlate`** (*imgOriginal*, *matchingChars*)
Extract plate image using contours/bounding box

`FindPlates.`**`findPossibleCharsInScene`** (*imgThresh*)
Similar to detectPlatesInScene(originalImage) except that it looks for characters in each rectangle.

1.8 flattened_images.txt

Training data for personal OCR

1.9 LPR_Final.py

`LPR_Final.`**`drawRedRectangleAroundPlate`** (*originalImage*, *truePlate*)
Draws a rectangle around the found plate. Uses tuples of points to make each side of the rectangle.

`LPR_Final.`**`main`** ( )
Main function. Calls other functions/files.

`LPR_Final.`**`writeLicensePlateCharsOnImage`** (*originalImage*, *truePlate*)
Writes characters on top of input image. Also includes rectangle around plate.

1.10 PossibleChar.py

class `PossibleChar.`**`PossibleChar`** (*_contour*)

Self.contour Holds the contour for each char in scene

Self.boundingRect
Holds the bounding rectangle points for each char in scene

Self.intBoundingRectX
Used for distanceBetweenChars() = abs(firstChar.intCenterX - secondChar.intCenterX)

Self.intBoundingRectY
Used for distanceBetweenChars() = abs(firstChar.intCenterY - secondChar.intCenterY)

Self.intBoundingRectWidth
Width of possible character. Used in conjunction with intBoundingRectX to get X position

Self.intBoundingRectHeight
Height of possible character. Used in conjunction with intBoundingRectY to get Y position

Self.intBoundingRectArea
Area of possible char

Self.intCenterX Used mainly for distance and angle between chars

Self.intCenterY Used mainly for distance and angle between chars

Self.fltDiagonalSize

Used to find matching chars

Self.fltAspectRatio

Used to check if rectangle contains possible char

1.11 PossiblePlate.py

class `PossiblePlate.`**`PossiblePlate`**

Self.imgPlate original image

Self.imgGrayscale

grayscale image

Self.imgThresh thresholded image

Self.rrLocationOfPlateInScene

rectangle location in scene

Self.strChars plate value found

1.12 preprocess.py

`preprocess.`**`extractValue`** (*imgOriginal*)

Extracts HSV values from image

`preprocess.`**`maximizeContrast`** (*imgGrayscale*)

Does what it says on the tin. Maximizes contrast of the image.

`preprocess.`**`preprocess`** (*imgOriginal*)

Contains all preprocessing functionality. Converts input to grayscale -> max contrast -> blur with zeros -> gaussian blur -> threshold Returns grayscale and threshold images

d

f

l

p

P

R

S

T

W