Assignment 1

TDS3651 Visual Information Processing

Out: **December 11, 2016** (Week 4)
Due: **January 8, 2017** (Week 8)

1 Introduction

1.1 Objective

To design an algorithm that returns the value of all the coins in an image.

Of course, an image of a bunch of coins can overlap and pile up (as crazy as in Figure 1), but that would be an impossible task for a system to process it visually. In this assignment, we assume the coins to be lying flat on a surface. Think of some kind of coin inspection conveyor belt. At most, the coins will touch each other, or overlap *minimally*.

Figure 1: Malaysian coins

1.2 Guidelines

- This is an individual assignment.
- In regards to the objective of this assignment, you are **NOT** allowed to use third-party packages to assist you in this problem. Packages '`pip`'-ed from Python Package Index (PyPI) are acceptable.
  - **Important:** Do not upload your own packages or algorithms that you have created for this assignment to public repositories such as Github/BitBucket, which may allow others to "use" portions of these packages to solve the same assignment. (You can do so after the assignment is over). All assignments will be scrutinized in this aspect.
- You can use Spyder (which comes with iPython console), but you can use your own favourite IDE or opt for basic text editor & command line.
- This assignment is worth 20% of coursework marks.
- Late-Day policy applies.
- Submission deadline: January 8, 2017, 11.59PM (Sunday of Week 8). There will be 3 extra days before hard deadline.

2 Data

Figure 2: Sample coin images from the data provided

A set of 10 different coin images is provided for you to design the counting algorithm. The coin images consist of a combination of different Malaysian coins (of various versions, including the most recent coin design with the "gold-coloured" 20 cents and 50 cents). All images are captured from a fixed height but somewhat uncalibrated (it is not exact, there are some slight differences in height of the image captured). Figure 2 shows some sample coin images from the dataset provided. The following data is provided:

- The raw coin images (in .jpg format), named with the file name, `'coin_sX_Y.jpg'` where X denotes the coin set number; '1' for Set 1), and Y denotes the image number in the set. There are 10 coin images in a set.
- The pickle data file (`'coinset1.pkl'`) contains the pre-loaded 4-D array of all the coin images in the set (*i.e.* `height*width*3*10`), and the ground-truth information.

Do not tamper with the information in these data files or manually alter the images. All your submitted codes will be re-run with the original data!

3 Scripts and Functions

3.1 Codes to Write

Your working function `'coinCount'` that you need to write is contained within `'coinCounting.py'`

```
def coinCount(coinMat, i):

    # write your code here
    ...

    return ans
```

The inputs and output of the `'coinCount'` function are as specified as follows:

- `coinMat`: Coin set (RGB) images, in 4-D `numpy` array of `row*col*3*numImages`, where `numImage` denotes the number of images in coin set. This array is provided has been loaded in the evaluation scripts.
- `i`: Coin set image number (1, 2, . . . , 10)
- `ans`: Total value of the coins in the image, in float type

No visualization codes or functions were provided. You have to write your own for purpose of visualizing the outputs or to generate nice figure/plots for reporting.

3.2 Evaluation Functions

Two evaluation functions are provided to test your algorithm:

- `evaluateImg.py`: Evaluate a single coin image with its ground truth and return the result
- `evaluateSet.py`: Evaluate all coin images in the set with the ground truth and display the results in two styles

Both functions are runnable on Anaconda Prompt or standard command-line prompt (if necessary path settings have been configured). You can use the `'-h'` switch to get further help on how to use these functions, and what other options are there.

3.2.1 Package Requirement

The vanilla Anaconda installation does not come with the `PrettyTable` package. Please install via `pip` at Anaconda Prompt.

```
>> pip install prettytable
```

3.2.2 Example of Usages

This command evaluates image number 5 in the default coin set (`coinset1.pkl`):

```
>> python evaluateImg.py 5
```

To specify image number 3 from a different coin set *e.g.* `coinset2.pkl`:

```
>> python evaluateImg.py -i coinset2.pkl 3
```

This command evaluates the whole coin set (default set `coinset1.pkl`) on the simple result display setting:

```
>> python evaluateSet.py

Total error: $ 6.90
Accuracy: 20%
Code runtime: 4.8047 seconds
```

To show the detailed breakdown for each image in table format, add the verbosity switch `'-v'`

```
>> python evaluateSet.py -v

########  DETAILED RESULTS  ########
+-------+--------------+------------+--------------+
| Image | Accuracy Hit | Error (RM) | Run time (s) |
+-------+--------------+------------+--------------+
|   1   |      1       |    0.00    |    0.9436    |
|   2   |      1       |    0.00    |    0.3369    |
|   3   |      0       |    0.85    |    0.9069    |
|   4   |      0       |    0.50    |    0.1580    |
|   5   |      0       |    0.80    |    0.2087    |
|   6   |      0       |    0.85    |    0.0663    |
|   7   |      0       |    1.50    |    0.7128    |
|   8   |      0       |    0.85    |    0.5525    |
|   9   |      0       |    0.75    |    0.3579    |
|   10  |      0       |    0.80    |    0.5610    |
|       |              |            |              |
|  All  |      2       |    6.90    |    4.8047    |
+-------+--------------+------------+--------------+
```

To perform this full evaluation on another coin set, simply add the `'-i'` switch with the pickle data file name:

```
>> python evaluateSet.py -i coinset2.pkl
```

Alternatively, you can run these evaluation scripts from iPython console as well, using the `runfile` command. E.g.

```
In [1]: runfile('evaluateImg.py', args='-i  coinset2.pkl 3');
```

3.3 Evaluation Sets

There are 2 evaluation coin sets. One is released earlier for you to design your algorithm (`'coinset1.pkl'`), if you are able to get a good accuracy with this coin set, it's a job well done already!

The second set (`'coinset2.pkl'`) will be released one week before submission deadline, consisting of coin images captured under similar conditions as the first coin set, but with more variations and challenging scenarios.

4 Submission

Submit the following in a ZIP file via MMLS Assignment page:

- **Code**: `coinCounting.py` and all other additional support codes
- **Report (in PDF)**: Proposed outline:
  - Abstract (short)
  - Introduction (short)
  - Description of Methods
  - Results & Analysis
  - Suggestions for Improvement

Please **do not** submit anything in hardcopy (report) or in stored media (CD/DVD) form.

4.1 Mark Distribution

The following table shows the mark distribution for this assignment:

Code (10%)	Methods Used	4
	Creativity/Originality in Solution	2
	Visualizations	3
	Clarity/Readability	1
Report (10%)	Abstract (short) & Introduction	2
	Description of Methods	3
	Results & Analysis	3
	Suggestions for Improvement	2
Bonus (max. 2%) – for exceptional achievement in accuracy/run time results		

End of Assignment 1 Guideline
John See, 2016