

```
In [ ]: # importing necessary libraries
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import requests
import re
from urllib import request
```

```
In [ ]: # Getting the datasets
!wget https://raw.githubusercontent.com/SarthakV7/Kaggle_google_quest_challenge/master/test.csv
!wget https://raw.githubusercontent.com/SarthakV7/Kaggle_google_quest_challenge/master/train.csv

--2020-07-13 14:47:20-- https://raw.githubusercontent.com/SarthakV7/Kaggle_google_quest_challenge/m
aster/test.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.0.133, 151.101.64.133, 15
1.101.128.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.0.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 976980 (954K) [text/plain]
Saving to: 'test.csv'

test.csv          100%[=====>] 954.08K  ---KB/s    in 0.1s

2020-07-13 14:47:20 (7.56 MB/s) - 'test.csv' saved [976980/976980]
```

```
In [ ]: # Reading the data
train = pd.read_csv('train.csv')
train.head(2)
```

Out[]:

	qa_id	question_title	question_body	question_user_name	question_user_page	answer	answer_u
0	39	Will leaving corpses lying around upset my pri...	I see questions/information online about how t...	Dylan	https://gaming.stackexchange.com/users/64471	There is no consequence for leaving corpses an...	I
1	46	Url link to feature image in the portfolio	I am new to Wordpress. i have issue with Featu...	Anu	https://wordpress.stackexchange.com/users/72927	I think it is possible with custom fields.\n\n...	

For scraping data using external links I used 3 features that were provided in the initial datasets. 'url', 'question_user_page' and 'answer_user_page'.

In []:

URL

The link provided in this feature takes us directly to the question page on stackoverflow. On that page I found 3 useful features to scrape-

- The upvote count of the best answer (best answer = answer provided in the dataset) as 'upvotes'.
- The comments in the best answer. (named as comments_0 in the scraped feature dataset).
- The top 3 answers apart from the best answer and their comments. (named as answer_1, comment_1, answer_2, comment_2, answer_3, comment_3 in the scraped feature dataset).

```
In [3]: %matplotlib inline
from IPython.display import Image
Image('2.png', width=920, height=480)
```

Out[3]:

The screenshot shows a Stack Exchange question page. On the left sidebar, the 'Questions' tab is selected. A red arrow labeled 'Upvotes' points to a red box containing the number '27'. Another red arrow labeled 'Comments' points to a red box containing two comments. The question text is: 'Here and there I read of people who set their lens to a small aperture while testing their sensor cleanliness, supposedly to get the best image of dust speckles. However, the image of the on-sensor dust particles should not, to my understanding, be affected by the sharpness induced by the lens settings. Same is true for dust particle on the lens elements themselves. It makes me wonder - do these people misunderstand the theory of how the optical system work, or am I missing something?'. The question has tags: aperture, sensor, cleaning, dust. The asker is 'ahockley' with a reputation of 20.9k. The question was asked on May 16 '11 at 5:27. The first comment is by 'Jukka Suomela' on May 19 '11 at 10:06. The second comment is by 'ysap' on May 19 '11 at 13:43. On the right, there is a 'Photo of the Week' section featuring a photo of a car in a tunnel, titled 'Audi-Go-Round by Alaska Man'.

StackExchange Search on Photography... Join this community

Why use a small aperture when trying to see sensor dust?

Asked 9 years, 2 months ago Active 9 years, 2 months ago Viewed 4k times

27

Upvotes

4

aperture sensor cleaning dust

share improve this question follow

edited May 16 '11 at 5:41 asked May 16 '11 at 5:27

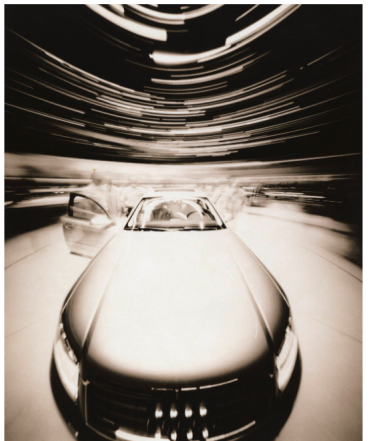
ahockley 20.9k 5 69 144 ysap 10.8k 3 31 55

1 Perhaps this question could be extended to cover other aspects: Does it matter if I use a wide-angle lens or a telephoto lens? A zoom or a prime? Defocus to infinity or very close? (At least with some quick experiments that I did, all other things being equal, dust was easier to spot with a *shorter* focal length – however, the combination of a short focal length and small aperture meant that the background was not defocused enough...) – Jukka Suomela May 19 '11 at 10:06

1 @JukkaSuomela - which is why, I guess, the best method will be to test aiming at clear blue skies. – ysap May 19 '11 at 13:43

add a comment

Photo of the Week



Audi-Go-Round by Alaska Man

In [6]: `Image('3.png', width=620, height=480)`

Out[6]:

Stack Exchange Search on Photography... Join this community

Home Questions Tags Users Unanswered

Why use a small aperture when trying to see sensor dust? Ask Question

Asked 9 years, 2 months ago Active 9 years, 2 months ago Viewed 4k times

▲ Here and there I read of people who set their lens to a small aperture while testing their sensor cleanliness, supposedly to get the best image of dust speckles. However, the image of the on-sensor dust particles should not, to my understanding, be affected by the sharpness induced by the lens settings. Same is true for dust particle on the lens elements themselves. It makes me wonder - do these people misunderstand the theory of how the optical system work, or am I missing something?

▼ 27 4

aperture sensor cleaning dust

share improve this question follow

edited May 16 '11 at 5:41 asked May 16 '11 at 5:27

ahockley 20.9k 5 69 144 ysap 10.8k 3 31 55

1 Perhaps this question could be extended to cover other aspects: Does it matter if I use a wide-angle lens or a telephoto lens? A zoom or a prime? Defocus to infinity or very close? (At least with some quick experiments that I did, all other things being equal, dust was easier to spot with a *shorter* focal length - however, the combination of a short focal length and small aperture meant that the background was not defocused enough...) - Jukka Suomela May 19 '11 at 10:06

1 @JukkaSuomela - which is why, I guess, the best method will be to test aiming at clear blue skies. - ysap May 19 '11 at 13:43

add a comment

2 Answers Active Oldest Votes

▲ 19 ▼

✓ If the dust was really on the sensor proper, you'd be absolutely correct.

At least in the normal case, it's virtually impossible for dust to get on the surface of the sensor itself, because there's a couple millimeters or so of filters directly in front of the sensor. The front-most of these is (at least in the usual case) the AA filer.

The important thing is that all of this transparent glass. Therefore, with a wider aperture, there's more light coming at the sensor from various angles. Since the light can travel at whatever angle through those filters (because they're all at least mostly-transparent glass), the dust spots won't normally block all the light. With a smaller aperture, the light comes nearly straight back from the small aperture, so the edges of any dust spots are clearly defined.

In practice, the difference is pretty obvious. Here's a shot at f/1.7, then shot taken a few moments later at f/22 (same camera, same lens, etc. -- all that's changed is the aperture and shutter speed):

f/1.7:

Photo of the Week

Audi-Go-Round by Alaska Man

Submit your photo Hall of fame

Featured on Meta

Feedback post: New moderator reinstatement and appeal process revisions

The new moderator agreement is now live for moderators to accept across the...

2020 Community Moderator Election - Failed Election

Linked

2 Question About the D600 Dirty Sensor Test - Why Use a High f-Stop?

26 Is it safe to change the lens on my new DSLR?

20 What should I look out for when buying a second-hand DSLR body?

```

In [ ]: # Here is the code. Since all of the urls are of stackoverflow, they have the same html hierarchy.
def get_answers(url):
    try:
        get = request.urlopen(url).read()
        src = BeautifulSoup(get, 'html.parser')
        upvotes, posts = [], []
        correct_ans, comments = [], []
        new_features = []
        post_layout = src.find_all("div", class_ = 'post-layout')
        l = len(post_layout)
        for p in post_layout[:l]:
            posts.append(p.find_all('div', class_='post-text')[0].text.strip())
            upvotes.append(int(p.find_all("div", class_ = 'js-vote-count grid--cell fc-black-500 fs-title g
rid fd-column ai-center')[0].get('data-value')))
            correct_ans.append(len(p.find_all("div", class_ = 'js-accepted-answer-indicator grid--cell fc-g
reen-500 ta-center py4')))
            comments.append('\n'.join([i.text.strip() for i in p.find_all('span', class_='comment-copy'))])

            idx = np.argmax(correct_ans)
            new_features.append(upvotes.pop(idx))
            new_features.append(comments.pop(idx))
            del posts[idx]
            if l < 3:
                k=1
            else:
                k=3
            for a,b in zip(posts[:k], comments[:k]):
                new_features.append(a)
                new_features.append(b)

            for a,b in zip(posts[:3-k], comments[:3-k]):
                new_features.append('')
                new_features.append('')

        return new_features

    except:
        return [np.nan]*8 # return np.nan if the code runs into some error like page not found

```

```
In [ ]: # Collecting data
        from tqdm import tqdm
        data = []
        for url in tqdm(train['url']):
            data.append(get_answers(url))
```

```
In [ ]: # Saving as dataframe
        columns = ['upvotes', 'comments_0', 'answer_1', 'comment_1', 'answer_2', 'comment_2', 'answer_3', 'comment_3']
        scraped = pd.DataFrame(lens, columns=columns)
        scraped.to_csv(f'scraped_posts.csv', index=False)
```

```
In [ ]:
```

Question_user_page, Answer_user_page

The link provided in this feature takes us directly to the user's page on stackoverflow. On that page I found 4 useful features to scrape-

- Reputation of the user.
- Number of gold points achieved by the user.
- Number of silver points achieved by the user.
- Number of bronze points achieved by the user.

These features are stored as reputation_q, reputation_a, gold_q, gold_a, silver_q, silver_a, bronze_q, bronze_a in the dataset (here 'q' corresponds to the data from question user page and 'a' corresponds to the data from answer user page).

In [8]: Image('1.png', width=920, height=480)

Out[8]:

Stack Exchange user: 1024

Photography

Home Questions Tags Users Unanswered

Profile Activity

10,778 REPUTATION

3 gold_score 31 silver_score 55 bronze_score

Canon EF-S 10-22mm f/3.5-4.5 USM
Canon EF-S 18-55mm f/3.5-5.6
Canon EF 28-135mm f/3.5-5.6 IS USM
Canon EF 50mm f/1.4 USM
Canon EF 70-200mm f/2.8L IS USM
Canon EF 75-300mm f/4-5.6 III

Flashes:
 Canon Speedlite 580EX II
 Canon Speedlite 380EX

Tripod:
 Manfrotto 3021BPRO
 Manfrotto 322CR2

More accessories and bags...

Communities (34)

Community	Score
Photography	10.8k
Stack Overflow	6.7k
Ask Ubuntu	5.9k

Top tags (295)

Tag	Score	Posts
lens	183	43
canon	106	30
equipment-recommendation	96	33

Meta user **Network profile**

226 answers **15** questions **~1.2m** people reached

Massachusetts
 Member for 9 years, 11 months
 971 profile views
 Last seen Jul 9 at 13:07

```

In [ ]: # code for scraping the data. Since all of the urls are of stackoverflow, they have the same html hierarchy.
train = pd.read_csv('train.csv')
def get_user_rating(url):
    try:
        get = request.urlopen(url).read()
        src = BeautifulSoup(get, 'html.parser')
        reputation, gold = [], []
        silver, bronze = [], []
        template = src.find_all("div", class_ = 'grid--cell fl-shrink0 ws2 overflow-hidden')[0]
        reputation = int(''.join(template.find_all('div', class_='grid--cell fs-title fc-dark')[0].text.strip().split(', ')))
        gold = int(''.join(template.find_all('div', class_='grid ai-center s-badge s-badge__gold')[0].text.strip().split(', ')))
        silver = int(''.join(template.find_all('div', class_='grid ai-center s-badge s-badge__silver')[0].text.strip().split(', ')))
        bronze = int(''.join(template.find_all('div', class_='grid ai-center s-badge s-badge__bronze')[0].text.strip().split(', ')))

        output = [reputation, gold, silver, bronze]
    except:
        output = [np.nan]*4 # return np.nan if the code runs into some error like page not found

    return output

from tqdm import tqdm
data = []
for url in tqdm(train['answer_user_page']):
    data.append(get_user_rating(url))

columns = ['reputation', 'gold', 'silver', 'bronze']
scraped = pd.DataFrame(data, columns=columns)
scraped.to_csv(f'scraped_score.csv', index=False)

```

In []:

I collected the data for both train and test datasets.

In []: