EE368/CS232: Final Project

Project report and source code due
Friday, March 13, 2020, 11:59 pm PT

Please respect the Stanford Honor Code. You may use the lecture notes, your own notes, books, scholarly papers, or any information available online, as long as your properly cite them in your report. Do not copy or paraphrase text without proper attribution. Do not copy or modify figures without proper attribution. Under no circumstances, manipulate, suppress, or make up experimental results. You are expected to work on the final project individually. However, you may discuss ideas with the teaching staff and the class by publicly posting on Piazza. Otherwise, please refrain from discussing the project until after the submission deadline.

Submission guidelines

- Submission must be made electronically on Gradescope. Please submit your answers as a single pdf file to the "Final Project" assignment, with relevant Matlab code and figures. Late submissions will not be accepted.
- If you experience issues submitting to Gradescope, please email your solution to the staff email address (ee368-win1920-staff@lists.stanford.edu).
- You are expected to submit a zip archive of all source code along with a concise report describing the image processing algorithm(s) you have developed, the reasoning behind the choices you have made, along with experimental results documenting the performance on the training set we provide.
- A typical written report will comprise 1000 words (2000 words max, not including references) along with graphs, pictures and references. Please use the IEEE conference paper template (*http://www.ieee.org/conferences_events/conferences/publishing/templates.html)*.

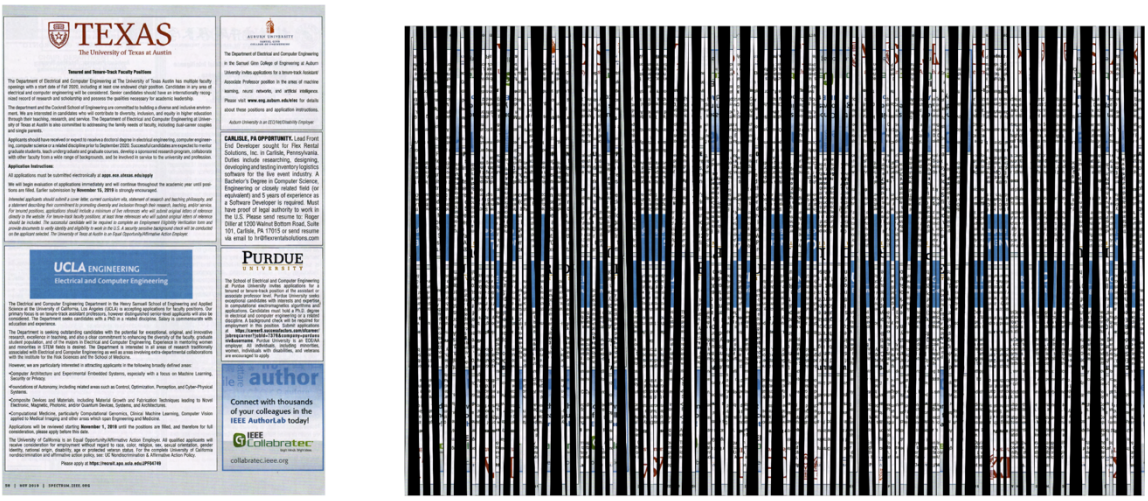
Assignment

Recovering shredded documents is a challenge occasionally faced by law enforcement. In this project, you will apply what you have learned to develop an image processing algorithm that can recover original pages from their shreds.

You are provided digital images of the paper strips resulting from shredding 20 pages randomly selected from different issues of the IEEE Spectrum magazine. The shreds have been pre-sorted into directories so that each directory contains shreds from a single page. Each page has been shredded into 100 narrow strips. The entire data are organized as `xx/yy.png`, where `xx` indicates the page number (ranging from 00 to 19) and `yy` indicates the strip number (ranging from 00 to 99). We will refer to these data collectively as the *training set.* Additionally, each directory `xx` also contains an image of the unshredded page for reference, named `original.png`. Note that the unshredded page is provided only for reference and may not be used as an input to your algorithm that reconstructs the page from the shreds.

The shreds are in random order. They are roughly vertical (within a few degrees), but might or might not be upside down. You may assume that no shreds are missing from any of the pages.

An example page and its corresponding shreds are displayed below.



Your code submission will be evaluated by the teaching staff on a *test set* containing the shreds of pages previously unseen but generally similar to the ones in the training set provided. Each of those test pages has also been shredded into 100 vertical strips in the same manner as the training set. *The test set does not contain the original, unshredded pages.*

The following criteria will be used for evaluation of your code submission.

1. The number of correctly matched seams. For example, if the correct order of the shreds is [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], but your code returns [2, 3, 4, 9*, 8*, 7*, 6*, 1, 5, 0], where the shreds marked with * are rotated by 180 degrees, your score would be: 2 correct seams in 2-3-4 plus 3 correct seams in 9*-8*-7*-6*. So, the total score will be 5 (out of 9). Since each page consists of 100 strips, the maximum score per page is 99.
2. The overall runtime of your code.
3. Any post-processing done on the stitched images to remove visible seams and artifacts.
4. Readability of your Matlab code based on structure and comments.

Your code must be submitted as a function block that is of the form

```
function [order, orientation, time, recovered_page] = assemble_shreds(shred_directory)
```

where (1) `shred_directory` is the directory containing the strips (in this case the directory named `xx`), (2) `order` is a vector giving the index of each strip (the `yy` value) in the order in which it should be stitched to reconstruct the page image; *the* `order` *vector cannot have duplicate indices*, (3) `orientation` is a binary vector that returns a 0 at location i, if the strip (yy = i) is to be used as is, and 1 if it is to be rotated 180-degrees, and (4) `time` is the total time it took for your entire function block to execute (Matlab functions `tic` and `toc`).

Please note:

1. It is critical that you follow the function call template described above since your code will be evaluated with an automated script. The exact naming of the variables is, of course, not important; however, the order in which the outputs are returned is!
2. Your algorithm must run on all pages without tweaks since the same code will be tested on all pages of the test set. To avoid overfitting to the training set, you might set aside some of the images provided and use them to test the robustness of our algorithm.
3. If your code results in a stitched page image that is rotated by 180-degrees, this still counts as a correct answer – there is no need to automatically rotate the stitched image into an upright position.