COMS W4735:Assignment 1
Due Feb. 14, 2023

This assignment is worth 20% of your course numeric score. It must be done individually and not in teams. A second and third assignment will also each be worth 20% of your course numeric score. A project proposal will be worth 7% of your score. The final project with demonstration and 10 page write-up will be worth 33% of your course score. There will be no exams.

1 Visual Combination Lock: Design Specifications

The goal of this assignment is to take a short sequence of visual images, and to determine from them if the user has placed some body part(s) in a predetermined sequence of poses ("what") and/or locations ("where"). For example, the program can require two images of the user's hand on a table, and the "lock combination" is if the hand is first placed with a closed fist in the center of the first image, then followed by a flat palm with fingers outwardly splayed (i.e., the gesture for "five!") in any of the four corners of the second. Many other variants are possible, and, in fact, part of the assignment (in the last step of four steps) will depend on how creative your domain engineering and your grammar are demonstrated to be.

To do this assignment, you will need access to a digital picture-taking device: a webcam, a scanner, a laptop with built-in camera, a cellphone with camera, a digital still camera, etc. The camera need not work in real time–you can store the pictures for later analysis. It is not necessary to buy such a device for yourself. But, you must take your own pictures, and transfer them to your own computer. Part of this assignment is deliberately left under-specified: you will have to find out how to best capture an image, which is usually not straightforward due to the combined effects of object surface, background, lighting, and camera particularities..

Size, color, resolution, etc., are up to you, and lighting, positioning, and background in particular will necessarily be learning experiences. Your choice of background that maximizes contrast to skin, and your choice of light position that minimizes shadows will probably turn out to be critical.

Also, you can use any language of your choice. Please see Courseworks Files in Week 0 for a list of the most common image processing packages for Python, C++, Java, and Matlab. Note that OpenCV is available for all languages, and traditionally has been the one which students have preferred. All of these do-it-yourself steps will help set you up for your own final project.

Because we do not have enough staff to allow live demos of this or the other two assignments, your documentation will be critical. Especially since your choice of hardware, operating system, language, and image package are entirely up to you, the course assistants will be relying on your *code documentation* and your *write-up*, rather than the execution of the system itself.

To help structure the assignment, it is broken down into four steps, with the full assignment worth the 20 points toward your final course score.

1.1 Step 1 (5 points) Domain engineering step.

First, to get a feel for what the images and the domain looks like, do the following.

If you are working with your own camera, you will need to have the appropriate drivers that dump camera input into files. You will also need to understand the format of these files, particularly if what you get is a compressed image. Please note that on most Unix machines, the command "convert", written up in the "man" pages, can help move images to and from the various formats. Most of the packages listed in Week 0 have utilities that will make this straightforward.

Once you have the ability to access the data of an image internal to your system's code, you should make sure your data is good. Capture an image of a hand under the lighting of your choice, against the background of your choice. To view it, you can use the program called "display" or others on Unix systems, or even just Media Player or Paint if using Windows. There are many other open source (or free) packages also available on the web, like IrfanView, GIMP, etc. And, most image processing packages make this possible by writing a very small program.

In general, you should capture your imagery and store them all in a directory for processing later. The system does not need to run in real time.

DELIVERABLES For step 1, you should describe the following.

1. How you captured the imagery: camera, lighting, objects, background. (Optionally, show a picture of your setup.)

2. Your environment: hardware, OS, language, packages.

3. Your library: how many images, the format they are stored in. what they were *intended* to capture,

4. And, if you stop at this Step, show a display of at least three images, in reduced resolution. (If you continue, the following Steps will require a display of images, anyway, so you don't have to put them in your write-up here, but the rest of the write-up is required.)

1.2 Step 2 (5 points) Data reduction step.

Find a way to manipulate the images to get a good binary image of the body part, by defining some region of the color space as "skin", which is fundamentally the color of blood.

You will also need some way of testing the image to see if it looks more like a fist or more like a splay. If you wish, you can clean up the image in various ways first, but good domain engineering should make much of that unnecessary. If you use OpenCV or other packages, there will be methods that make this easy, but you will need to read and understand their documentation, and then select and experiment with their (usually many) parameters.

To document this step, you need to have your system annotate each of the images you have taken in some way, indicating both the "what" and the "where". For example, "fist, upper right" or "splay, lower center", or "unknown, left", would be acceptable. One good way to do this is to have the system superimpose upon the image in a fixed place (say, at the very bottom or extreme right, where it lies fully in the background) some form of textual checklist. Most packages make this easy. Then, indicate within this checklist those attributes your system found, since this would also make explicit the vocabulary you are using for the grammar.

For example, "splay, lower center" could be annotated something like:

```
  fist          uppL     uppC     uppR
X splay         cenL     cenC     cenR
  palm          lowL     XXXX     lowR
```

DELIVERABLES For Step 2, make sure your write-up shows, in reduced resolution, each image in two ways: first as an actual image, and second as the binary intermediate.

1. Two images of a centered fist that are accurately recognized and labeled as "fist, center" (true positive)
2. Two images of a cornered splay that are accurately recognized and labeled as "splay, upper right" [or some other corner] (true positive)

Please note that the CAs will examine these images to ensure that they look like they were generated and analyzed by your code. If necessary, they may ask you during the grading for you to submit the actual image files themselves.

1.3 Step 3 (5 points) Edge cases and extension.

First, extend your library of images in the following way. Gather:

1. One image of a centered fist that is not recognized as "fist, center" (false negative)
2. One image that is incorrectly recognized as "fist, center" (false positive)
3. One image of a cornered splay that is incorrectly recognized as "splay, upper right" (false negative)
4. One image that is incorrectly recognized as "splay, upper right" (false positive)
5. Two images that are accurately recognized as "unknown", that is, neither "fist" nor "splay" (true negative)

Second, extend the "what" vocabulary to include "palm", which is midway between "fist" and "splay". Show the following, where the "where" is not important:

1. Two images of a palm that are correctly recognized as "palm". (true positive)
2. One image of a palm that is incorrectly recognized as "fist" (false negative)
3. One image of "splay" that is incorrectly recognized as "palm" (false positive)
4. Two images that are accurately recognized as "unknown", that is, neither "fist" nor "splay" nor "palm" (true negative)

DELIVERABLES For Step 3, show all the above images (there are 12 in all) in reduced binary form. And, for each false response describe why it appears that the "what" or the "where" or both were incorrectly determined.

1.4 Step 4 (5 points) Evaluation step.

You will do this step twice: once by yourself, and once by your friend.

First define some combination that are a sequence of three states. For example, <"fist, center", "splay, upper right", "palm">. Then show your system three *new* images that you have captured, and have the system say whether they would open the lock (while, of course, printing out a trace of the "what", "where" determination of each image.)

Using your experience in the previous steps, determine the following:

1. An easy sequence, that is, one that is easy to describe and which the system is most likely to recognize properly
2. A difficult sequence, that is, one that is complicated to describe and for which the system will often fail, either because the "what", or the "where", or both tend to be errorful in your system.
3. A "good" sequence, which is interesting and often works. This is the "sweet spot" between "interesting" and "doable".

Then, once you have determined these three sequences, find a friend. Explain the assignment to the friend, demonstrate your system on those three sequences, and ask the friend to repeat them.

DELIVERABLES For Step 4,

1. Explain how you determined the three sequences
2. Display the (new) reduced resolution binary intermediates you gathered from you.
3. Display the (new) reduced resolution binary intermediates you gathered from your friend.
4. Report on your system's overall success rate in terms of accuracy.
5. Report any feedback from your friend about the system: ease of use, confusability of "what", and of "where", and any other comments that could be used to improve the system.
6. Given your experiences, summarize how you would approach System 2.0: what you would keep, what you would change, etc.

Your grade for this step will depend in large part on your analysis and discussion. Please note that the prof is not responsible for the any impact of your system on your friendship.

2 General rules

Please note that whatever you do in code or in write-up, style counts. It is your obligation to write it all up so that the instructor and/or the course assistant can understand it on the *very first* try. You should submit only one zipped file, which incorporates within it (1) a narrative document describing your design approach and your choices of algorithms, imagery, testing, and analysis,

and (2) your internally-commented code, This format will also be the same one required for the final project: narrative plus code.

Please put all your submission files in one directory (named, for example,"HW1"), and then compress the contents of that directory into one file. Submit that compressed file to Courseworks. The name of that compressed file should be "myUNI_HW1", and it should have the appropriate extension like ".zip", except that "myUNI" should of course be replaced with your real Columbia UNI. *Please* use this convention as it makes the course assistants' job much easier: You may submit electronically as often as you wish, but it will be your *last* electronic submission before the deadline which is the official one that will be graded.

Please use Ed Discussion, early and often. It is a good way to explore and learn, from and with the rest of the class. And, you can use it anonymously, so you don't have to be afraid of looking stupid or unprepared. But, please start the assignment early, so that you can give yourself enough time to get any help you need.

Three final important notes about academic honesty:

1. Please read the department's statement of its policies and procedures, available on Courseworks. A reminder: this assignment is to be done individually, not as a team.

2. Any code that you did not write yourself has to be *documented* with a statement about its *source* and an explanation of why you have *permission* to use it.

3. We will be using software to help detect any cheating.