Class Project for the course ES297: Principles of Intelligent Systems

Project Report – 2nd Review

Location finding and Scenario Description Laptop application for blind navigation on the first floor (Library floor) of IIITB's main building

Using Wireless Signal Strengths

Method

First, a database of locations and their wireless signal strengths was created. For this, the user's location is manually given as input to the program. About 15 samples of signal strength measurements at intervals of 5s each, are taken and averaged. For each location, the location name, along with the set of access points and their strengths is stored, in descending order. All of these are stored in a single text file.

During testing, the signal strengths are measured and sorted, and the Euclidean distance (RMS value) is calculated for the signal with the values in the file. In addition to this, a simple ranking system is used.

The highest three strengths are compared with the database, and if a location has the same three highest signal strengths with the same ordering, it is ranked highest. A location with the same signal strengths, but different ordering is ranked next.

The location with the highest rank, and lowest total RMS value of the difference of the signals is given as the best estimate of where the user is.

Implementation

The code to do the above tasks was written in Python. Python was chosen because of its ease of use with convenient data structures such as lists and dictionaries. Also, modification is easier.

To collect signal strengths, the `nmcli` command was used. All location names along with their signal strengths were stored in a single text file in a suitable format.

Problems and Improvement

- The `nmcli` command does not seem to give updated values immediately, despite using the instructions given in the `nmcli` man page - “Be aware that nmcli is localized and that's why the output depends on your environment. It's important to realize that especially when you parse the output.” Call `nmcli` as `LC_ALL=C nmcli` to be sure the locale is set to "C" while executing in a script. Hence the previous location is printed for awhile, till the `nmcli` command updates itself. The old values need to be flushed out each time the command is run during testing and while the user is moving.
- The program gives the wrong location if an access point router that was switched on while forming the database, is switched off during testing. The dependence of the program on certain access points needs to be minimised. A very large database is probably required.
- Fluctuating signal strengths. This happens because of various reasons such as people present, number of other users, medium etc.
- Image recognition is used to overcome some of these problems.
- The program can be extended to accept clues from the user to improve its abilities, ie it can be made to “learn”.
- The program should be able to give error messages (“Sorry, can't help!”) if it is in a location it is not designed to recognise (such as the ground floor or 3rd floor).

Using Image Recognition

Similar to wireless strengths, a database of images is made, with each object/location having a set of 3 images shifted in position or orientation.
A simple correlation technique has been used to compare images.

Algorithm

1) Get RGB image as an array
2) Convert RGB to grey-scale using W3C luminance
3) Normalize data
4) Perform an accurate 2D correlation

Relative Luminance

Relative Luminance is the relative brightness of any point in a colorspace where 0=black and 1=white
E'Y= 0,299*E'R 0,587*E'G + 0,114*E'B

Normalization

data - data.mean()) / data.std())
Normalization is done to remove effects of lighting and exposure conditions on the images. The pixel intensity values are brought to a similar range of grayscale, which will make recognition of images easier and more accurate

2D Correlation

The 2D Correlation performs 2D correlation on two input matrices. Both linear and circular correlation can be computed. Two computation methods are available: a fast algorithm based on FFT and an accurate method based on shift accumulation. With a normalized result, it is easier to tell the degree to which the two input signals are correlated.
The run-time for the program using the above method was found to be too long for 1024*1024 size images. Hence small images were taken, and further scaled down to obtain a considerable improvement in speed.
An image with high correlation value (> 10000) is taken as a “match”.

Implementation

The code to perform image comparison was written in Python. The SciPy Library was used to perform image manipulation, correlation and other functions. To take a photo using the webcam in real time, the streamer package of Ubuntu was used.
The picture is taken by the webcam through a shell script and saved to a file, after which the python script runs to compare the image to those in the database. Once a match is found, a text message is printed. This text message can be easily converted to voice output (not yet done).

Problems and Improvement

- The time taken to find correlation values between all the images is quite large, despite scaling.
- The time of execution can be brought down by getting more information about the images and being able to exclude some images before performing correlation.
- Some pre-filtering can be used to exclude some images.
- Pixelisation has also been said to be useful.