**Object detection**

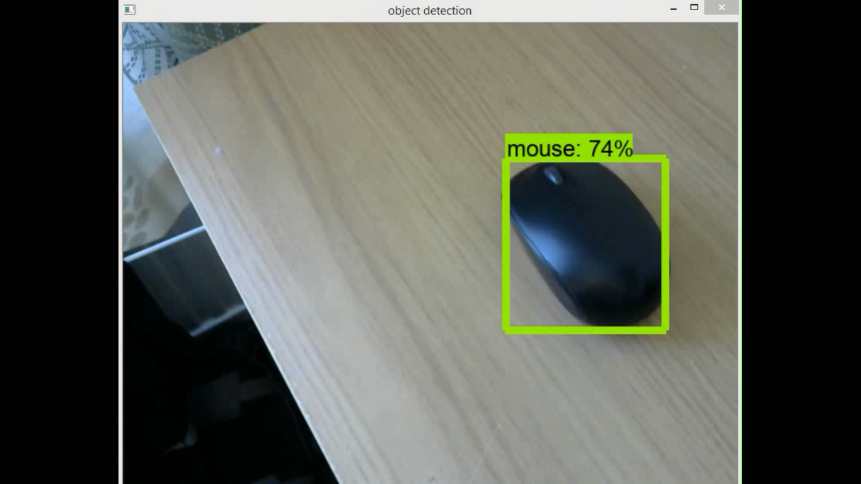
This project was developed with the intention of using a pre-trained model to detect everyday objects, and then generate an audio output stating the name of the object as well as the relative position of the object in the cameras field of view. Currently, the project used a webcam, however it could be further adapted to use a small camera which could be mounted onto a pair of glasses. Thus assisting visually impaired individuals with locating objects.

**Pre requisites:**

The program relies on a few pre-installed packages in order to work. The tensorflow module is utilised in order for the object recognition to take place using a pre-trained model. A ‘frozen\_inference\_graph’ is the actual model containing the ‘knowledge’ of the objects. The graph is only trained on a set number of object classes, therefore has a limited number of objects it can recognise.

In order for the text to be converted to audio, a ‘Text to speech’ engine is also used.

**How does it work?**

* The program makes use of threading to run two python programs at the same time. Parallelism is vital here as the module concerned with detecting the objects needs to run continuously, and so does the audio generating and playback module.
*  Each frame from the video feed will be analysed against the model for any objects it might be able to recognise. If none are present, then the next frame is looked at. If an object is recognised, a boundary box is placed around the object on the video feed. Additionally, the name of the object and the relative position of it is saved in a text file.
* Simultaneously, the other module will use the text to speech engine to convert the name of the object to an audio file. It will also add whether the object is to the ‘right’ or ‘left’ of the user. The mp3 file generated will then be played out.



**Limitations:**

* The program will only be able to detect objects the model has been trained for. Any other objects will just fall under the radar. However, this can be rectified by training one of our own models will object specific to the user’s requirement, thereby rendering the program much more efficient and useful.
* Under low light conditions, the program will fail to detect objects, even if it has been trained to do so, simply because, not much details of the object can be made out. Instead, an infared camera can be used which will allow objects under low light conditions to still be seen, however this may mean that the model would also have to be re-trained.
* Due to the visual processing requirements of the program, it is very CPU intensive. As a result, if multiple objects are detected one after the other in quick succession, the related audio output might be somewhat delayed. An algorithm to filter out these ‘unnecessary detections’ could be used utilised to reduce the latency.
* The model is also not always 100% accurate as can be seen with the video where the bottle is mistaken for a vase.