# Abstract

This thesis aims to create a framework for the development and optimisation of low-latency video encoding and transmission.

The main problem is to understand the sources of delay, and measure them accurately to assist in finding ways, so to reduce them enough for being applied to real time applications, such as remotely piloted vehicles.

The first method studied to analyse the delay in a video streaming transmission was to stream the video of a stopwatch (captured by an external webcam) and to capture a picture of the time shown on the decoder window. The video encoder software used was FFmpeg, whilst MPlayer was used as a low-latency decoder.

Initially, after various attempts, the encoder was configured to transmit the video with only 200 millisecond of delay, which was measured by comparing the time of the stopwatch with the time of the stopwatch displayed by MPlayer.

The measured delay was the end-to-end delay, including the encoder, UDP transmission and decoder delays. The result was not considered satisfactory because the delay was excessive.

Another method was developed to calculate the end-to-end delay, which was to stream the computer screen displaying a real time clock with FFmpeg, displaying the MPlayer output on the same computer screen, and then grabbing the screen showing both clocks. Comparing the two times allowed us to measure an average end-to-end delay of 142 millisecond, which included the encoding and decoding time.

The next step was to examine the delay of every single component of the video transmission chain.

To develop a tool to allow us to do this, an attempt was made to modify the FFmpeg code in order to also send the time when the packets leave the encoder with the transmitted packets, in order to allow us to calculate the encoder delay. This approach failed because of the difficulty to find the part of the code to modify.

Another attempt was then made to configure FFmpeg to output both a coded and a raw video format, to display the video after being processed by the encoder but before being UDP transmitted on the network, in order to compare the encoder timer with the end-to-end time. That solution didn't work because the raw video increased the delay and the result was that the latency of the raw video was more than the total delay.

The best solution developed, was to use Microsoft Windows Environment, display a real time clock, grab and transmit one frame showing the time with FFmpeg, and then use the MATLAB *judp* library to capture the time of the first UDP packet of the transmitted frame. By subtracting the

time of the capture timer from the time grabbed by MATLAB, it was possible to measure the encoder delay, which was found to be 80 milliseconds for the first Intraframe.

To perform a deeper analysis, it was decided to find the combined time of the first I-frame and the first P-frame. The method used was to transmit two frames instead of one and use the Wireshark network analysis tool to obtain a more accurate timing of the UDP packets. By examining the UDP frames grabbed by Wireshark, it was possible to find the time of the I-frame and of the P-frame.

To find the network delay we used the same methodology to measure the encoder latency, but using two different computers to send and receive the signal.

Finally, the decoder delay was found by the difference between the end-to-end delay, and the encoder and network latency found in the analysis described above.