Variable bit rate MPEG video implementation for drone applications

by Christian K. Laustsen, Alexander G. Brandi and Martin L. Madsen

Problem and requirements

- Poor signal quality in wireless networks
  - Loss in video frames
  - Insufficient video feed quality (stuttering)
- Temporary loss of connection
- Variable bit-rates can help solve problems in video streaming over poor quality networks
  - A requirement of 300Kb/s - 10Mb/s

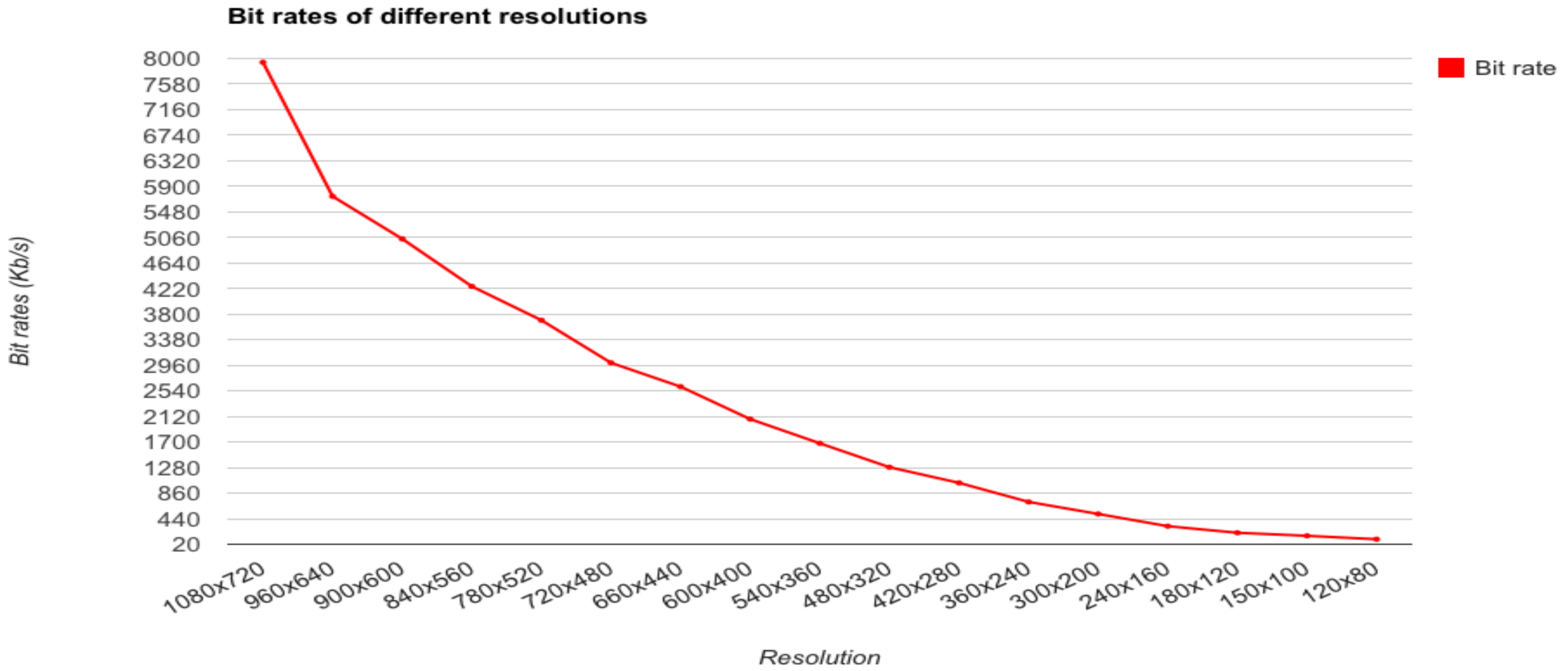
Using ffmpeg

- Transcoding to different resolutions
- Lowering the FPS
- Converting to a gray scale

Transcoding to different resolutions

- Convert to a set of resolutions using ffmpeg
    - ffmpeg -i NoAudio.mov -crf 17 -vf scale=width:-1 Output.mov
- Original resolution: 1080x720
- Transcoded to: 1080x720, 960x640, 900x600, 840x560, 780x520, 720x480, 660x440, 600x400, 540x360, 480x320, 420x280, 360x240, 300x200, 240x160, 180x120, 150x100 and 120x80

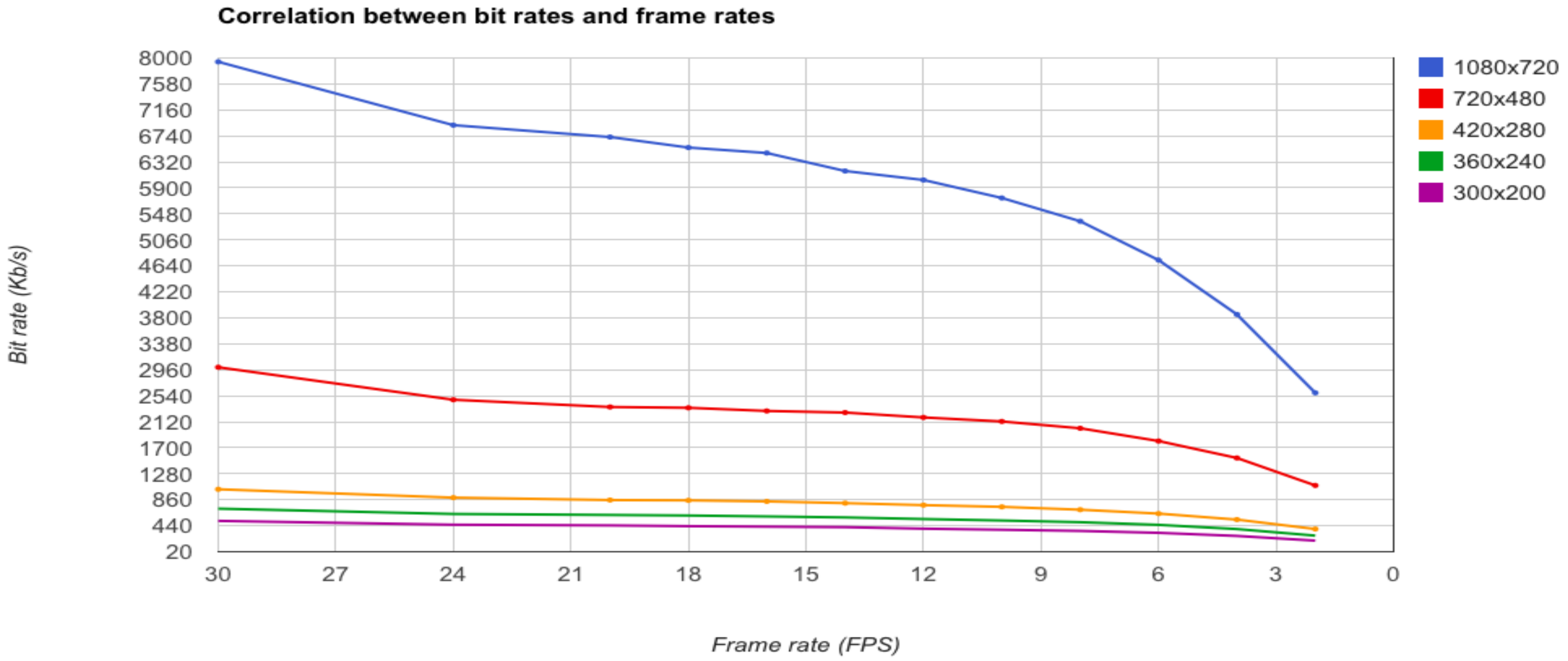
Transcoding to different resolutions



Lowering the FPS

- Lowering the FPS with ffmpeg (only on a subset of the scaled resolutions)
    - ffmpeg -i ScaledInput.mov -crf 17 -r fps Output.mov
- Original fps: 30
- Transcoded to: 24, 20, 18, 16, 14, 12, 10, 8, 6, 4 and 2

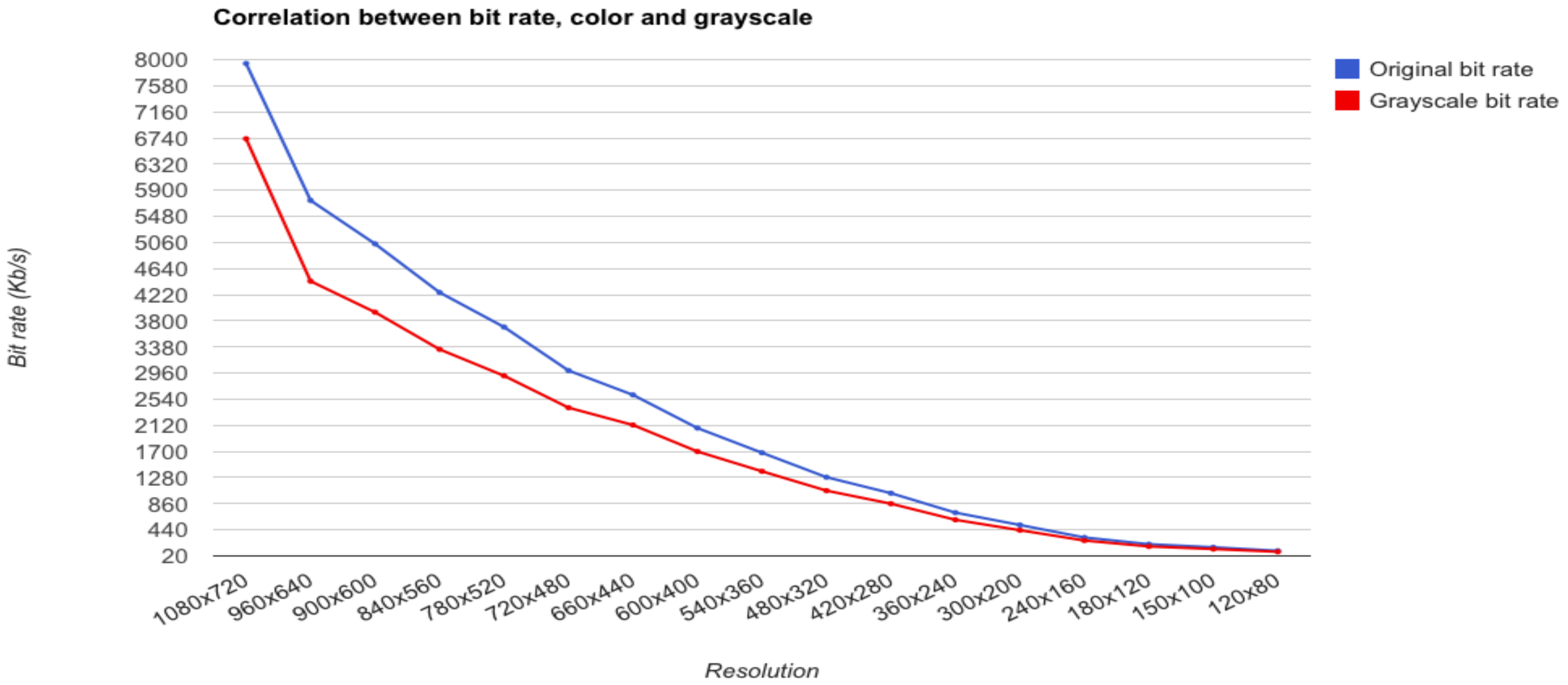
Lowering the FPS



Converting to a gray scale

- Removing colors using ffmpeg
  - ffmpeg -i ScaledInput.mov -crf 17 -vf hue=s=0 Output.mov

Converting to a gray scale



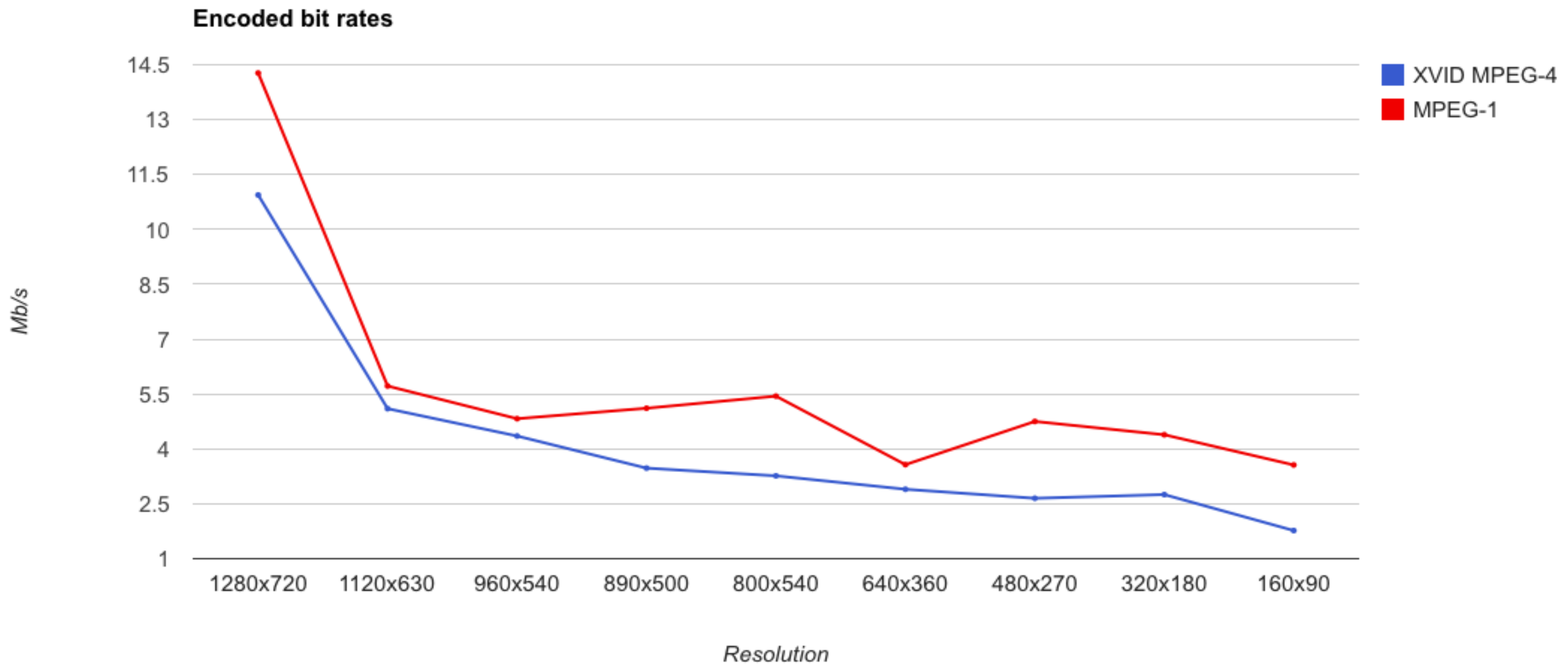
Comparison

- Lowering resolution
  - Most effective
  - Quickly decreases quality, especially below 420x280
- Lower FPS
  - Less effective, but still saves some bits
  - Can go down to around 10 FPS while still keeping a decent quality
- Removing colors
  - Not very effective
  - Doesn’t affect the quality (except if you need colors of course)
- Common for the last two, is the decrease in efficiency as the resolution goes down

Python implementation

- Prototype implementation of a server sending a RAW video stream, and a client displaying the stream. Uses OpenCV (http://opencv.org).
- Server
    - Can alter the resolution it is sending with
- Client
    - Automatically adjusts to the received resolution
    - Can encode the video with different codecs (which is normally done on the server)
- Demo

Python implementation



Android drone application

- Uses FFmpeg library
- Custom JNI libs (pre-compiled .so files)
- Transcodes RAW data in different resolutions
    - 1920x1080, 1280x720, 1024x680, 640x480 and 320x240
- Non-connection oriented datagram packets
- Demo

Conclusion

- Variable MPEG bit-rates are possible by altering one or several parameters, such as resolution, frame rate and color depth.
- A suggested set up that fits the requirements, from the data gathered
  - Specs of low end video stream, ~325 Kb/s
    - Resolution: 300x200
    - Frame rate: 6 fps
    - Full colors
    - Low quality, but watchable
  - Specs of high end, ~7938 Kb/s but should be larger
    - Resolution: larger than 1080x720, since that is still below 10Mb/s
    - Frame rate: 24-30 fps (can be traded for more resolution)
    - Full colors
    - High quality

Further work

- An objective measurement of the quality, by f.x. calculating the PSNR
- In the python streamer client, decouple the displaying of the images, with the receivement of the frames
- Further development of the android implementation
- Decision on buffering video if no Internet connection is available
- Updating the report