H.264 Video Encoding for Lightweight Decoding

Students:
Dudu Asulin
Ido Radai

Supervisors:
Hilla Madar
Naama Hait

The Project Goal

- Reduce decoder complexity by using computational-aware mode decision scheme at the encoder side.
H.264 Encoder

- Exploits the temporal redundancy in video content – sections of the previous picture generally move a few samples.

- Use of Motion Compensation (MC) and Residual coding increases the efficiency of the video coding.

Motion Estimation (ME)
Improvements in H.264 ME

- Fractional-sample-accurate Motion Vectors (MV’s) – up to 0.25 pel resolution.

- Variable block sizes – mode decision.

The problem

- These improvements significantly increase the computational power needed to decode H.264-coded video.

- Many sub-pixel interpolations needed to calculate reference blocks values.
Interpolation calculation 0.5/0.25 pel

\[ b = \frac{(E - 5 \cdot F + 20 \cdot G + 20 \cdot H - 5 \cdot I + J) + 16}{32} \]

Proposed solution

- Consideration of expected decoding complexity is inserted into the encoding process.

- This enables the encoder to make decisions that will generate a bitstream that is easier to decode.
The solution – Part I

- Conventional ME uses Rate-Distortion optimization cost function:
  \[ J(m, \lambda_{\text{motion}}) = \text{SAD}(s, c(m)) + \lambda_{\text{motion}} \cdot R(m) \]

- Suggested cost function penalize complex Motion vectors:
  \[ J'(m, \lambda_{\text{motion}}, \lambda_{\text{MEC}}) = J(m, \lambda_{\text{motion}}) + \lambda_{\text{MEC}} \cdot C_{\text{ME}}(m) \]

\[ C_{\text{ME}}(m) \]

The solution – Part II

- Another Lagrangian cost function is used to choose between the different modes:
  \[ J(m, \lambda_{\text{MODE}}) = \text{SSD}(s, r) + \lambda_{\text{MODE}} \cdot R(s, c, m) \]

- Suggested cost function considers the complexity of all MV involved in the mode:
  \[ J'(m, \lambda_{\text{MODE}}, \lambda_{\text{MDC}}) = J(m, \lambda_{\text{MODE}}) + \lambda_{\text{MDC}} \cdot C_{\text{MODE}}(m) \]

\[ C_{\text{MODE}}(M) = \sum_{i=1}^{\text{num of MVs}} C_{\text{ME}}(m_i) \]
The implementation

• The presented solution was successfully implemented into the H.264 encoder developed by Nokia.

• Choosing the optimal values of \( \lambda_{MDC} \) & \( \lambda_{MEC} \) for different sequences proved to be a critical aspect of the suggested solution.

• We found that, generally, higher values are required for more complex sequences.

Choosing \( \lambda_{MDC} \) and \( \lambda_{MEC} \) values

• After several suggestions, we selected the following method for choosing the values.

• The values are calculated for each frame, in order to adapt to changes in the sequence.
Choosing $\lambda_{MDC}$ and $\lambda_{MEC}$ values

- Empirical relation that we used in order to reduce the search space: $\lambda_{MEC} = \sqrt{\lambda_{MDC}}$

- MAD used as an indicator of the frame complexity.

$\lambda_{MEC} = f(MAD, \chi)$

**Mean of Absolute Differences**

Parameter defined by the user (Default: 3)

---

Encoder performance

- Little to no penalty was added to the encoder as a result of our changes.

- Encoding time of most sequences was similar, with maximum penalty of under 3%.
Results

- **Foreman – QCIF (Bitrate – 50Kbps)** Initial PSNR: 32.41

![Decoding Time improvement - Foreman](image)

- **Coastguard – QCIF (Bitrate – 50Kbps)** Initial PSNR: 29.59

![Decoding Time improvement - Coastguard](image)
Results

- Bugs – QCIF (Bitrate – 100Kbps)  Initial PSNR: 24.14

![Decoding Time Improvement Chart - Bugs](image)

Suggestions for future work

- Further optimize the selection of model parameters for higher resolution video sequences.

- Different complexity models can be implemented, such as hardware-based complexity model.

- Further decoding time can be saved by disabling the in-loop de-blocking filter.
References


