SI/TI Tools

VQEG Meeting, Dec. 2021

Lukáš Krasula <l.krasula@gmail.com>
Cosmin Stejerean <cstejerean@fb.com>
Werner Robitza <werner.robitza@gmail.com>
SI/TI Overview

- Defined in ITU-T Rec. P.910
- Classify spatiotemporal complexity of video sequences
- Definitions:
  - SI: Standard deviation of Sobel-filtered image
  - TI: A basic motion difference feature for adjacent frames

\[
SI = \max_{time} \{ \text{std}_{space}[\text{Sobel}(F_n)] \}
\]

\[
M_n(i,j) = F_n(i,j) - F_{n-1}(i,j)
\]
Updates to SI/TI Functions

● Make SI/TI future-proof:
  ○ Harmonize existing implementations wrt. handling of edges and full/limited range conversions
  ○ Handle content > 8 Bit per channel
  ○ Handle HDR content
  ○ → provide an update for ITU-T Rec. P.910

● Provide an even better encoding complexity metric:
  ○ Proposal by Ioannis Katsavounidis: Enhancing SI/TI with basic motion compensation
  ○ Code is available and will should be shared soon
Activities so far

● New software developed:
  ○ Written in Python (slow, but good experimental platform)
  ○ Main branch for “stable” version with the “good old” SI/TI:
    https://github.com/Telecommunication-Telemedia-Assessment/siti-tools/
  ○ “Siti2020” branch for current updates:
    https://github.com/Telecommunication-Telemedia-Assessment/siti-tools/tree/siti2020

● Liaison statement to ITU-T Study Group 12, informing them of our activities and inviting contributions
Software Features

● Command-line interface and API
  ○ Select bit depth, HDR mode, HDR conversion functions and parametrization

● Read y4m container format
  ○ Chosen for simplicity/interoperability
  ○ Support for > 8 Bit content is experimental

● Treat frames according to conversion pipeline
  ○ Pipeline proposed by Lukáš Krasula, including implementations of all conversion functions

● Calculate SI and TI with “good old” functions

● Output to JSON
  ○ Allows re-using the settings from previous runs to enable reproducible results

● Output to CSV
  ○ For quick analysis
Demo time!
Special wrapper for Python-ffmpeg binding to get actual values without grayscale conversion.

YUV input → Y component extraction → Y → Normalize between [0, 1] → Y<sub>norm</sub> → Y<sub>norm'</sub> [0, 1] → Which HDR mode?

- HDR10: Apply EOTF from ITU-R BT.1886
- SDR: Apply EOTF from ITU-R BT.2100

Which HDR mode?

- HLG: Apply OETF<sub>PQ</sub> from ITU-R BT.2100
- SDR: Apply OETF<sub>PQ</sub> from ITU-R BT.2100

Values are now in display luminance

Depending on bit depth, input can be [0, 255], [0, 1023], ...

SI function (as-is) → Normalize back to 8-bit range → SI → SI results

This assumes a display with luminance from 0.1–300 nits

E.g., for 8 bit, convert [16/255, 235/255] to [0, 1]

Now calculated in PQ domain

I.e., multiply by 2^8-1

This assumes a display with luminance from 0.1–300 nits
Evaluation

New vs. “legacy” SI/TI on AVT-VQDB-UHD1
10s, 2160p, 8-bit SDR

→ Values are lower due to shift to PQ domain
Is it linear?

Z-normalize SI/TI scores per group (legacy/new), i.e. subtract mean, divide by standard deviation

Plot correlation and density (colored) for SI

→ Not just a linear shift
Open issues

- Performance
  - Python implementation a bit slow
  - Possibility to use fixed-point calculations and native C extensions
  - Buffered file reading with parallel computation of SI and TI values (with queues)
  - ...

- Range of SI values
  - SI values become lower after running OETF for SDR content, because you are limited to lower range of PQ domain

- Check PU21 domain (as an alternative to PQ)

See:
https://github.com/Telecommunication-Telemedia-Assessment/siti-tools/issues