Adventures in Implementing SI and TI

Cosmin Stejerean, Boris Baracaldo

December 18, 2020

Motivation

- Integrate SI and TI calculations into an ffmpeg filter to allow processing as part of a larger video processing pipeline
- Validate numerical accuracy of new filter
- Contribute internally developed ffmpeg filter as an open-source implementation of these features

Problems Encountered

- No publicly available test vectors to validate implementation
- Cross validation against other publicly available implementations resulted in significant differences
- Review of existing implementations found several inconsistencies
- Standard definition not provide sufficient clarification to determine which is correct

Definitions of frame level SI and TI features

Source: *Subjective video quality assessment methods for multimedia applications*, document ITU-T P.910 Recommendation, 04/2008

SI

The spatial perceptual information (SI) is based on the Sobel filter. Each video frame (**luminance plane**) at time *n* (F_n) is first filtered with the Sobel filter [$Sobel(F_n)$]. The standard deviation over the pixels (std_{space}) in each Sobel-filtered frame is then computed.

TI

The temporal perceptual information (TI) is based upon the motion difference feature, $M_n(i, j)$, which is the difference between the pixel values (of the **luminance plane**) at the same location in space but at successive times or frames. $M_n(i, j)$ as a function of time (n) is defined as:

$$M_n(i, j) = F_n(i,j) - F_{n-1}(i,j)$$

Treatment of boundary conditions

As specified by the standard definition

SI

Detailed Sobel definition is provided in Annex A of ITU-T P.910, section A.1

“The calculations are performed for all $2 \leq i \leq N - 1$ and $2 \leq j \leq M - 1$, where N is the number of rows and M is the number of columns. “

TI

Motion difference value at frame *n* is defined based on difference to frame *n-1*

$$M_n(i, j) = F_n(i,j) - F_{n-1}(i,j)$$

Motion difference does not appear to be defined for the first frame in the sequence

How do we extract the luminance plane?

And why does that matter?

- Raw pixel values in Y plane of a YUV representation?
- Convert full color representation to grayscale pixel format?
- What is the allowed range of values?
  - Limited/TV range of 16-235?
  - Full/PC range of 0-255?
  - 10-bit sources?
- Different choices result in different output values of the SI and TI calculations

A basic implementation for illustration purposes

Tested in MATLAB R2020b

```
function [E] = sobel(plane)
    k = [1 2 1; 0 0 0; -1 -2 -1];
    H = conv2(double(plane), k, 'valid');
    V = conv2(double(plane), k', 'valid');
    E = sqrt(H.*H + V.*V);
end

function [si] = si_feature(y_plane)
    S = sobel(y_plane);
    si = std(S(:));
end

function [ti] = ti_feature(previous_y, current_y)
    pixel_diff = double(current_y) - double(previous_y);
    ti = std(pixel_diff(:));
end
```

A basic implementation for illustration purposes

Tested in MATLAB R2020b

```
function [output] = siti(frames)
    previous_frame = [];

    for ii = 1:length(frames)
        frame = frames{ii};
        if isempty(previous_frame)
            previous_frame = frame;
        end
        si_values(ii) = si_feature(frame);
        ti_values(ii) = ti_feature(previous_frame, frame);
        previous_frame = frame;
    end

    output = horzcat(si_values, ti_values);
end
```

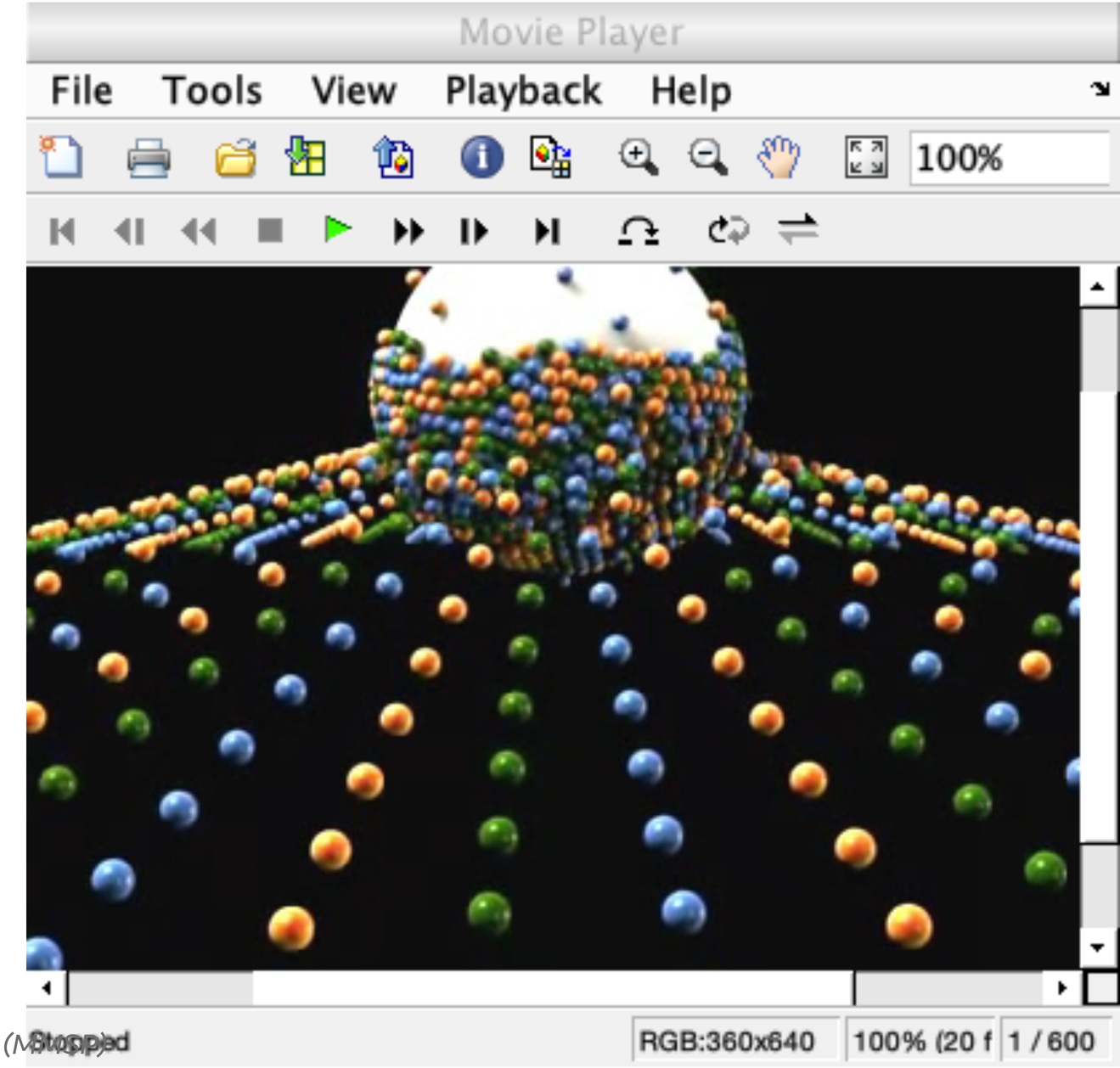
Implications of luminance range

Process first 10 frames of the following video

Sample clip from the YouTube UGC dataset

available under Creative Commons license

https://media.withyoutube.com/



Balu Adsumilli, Sasi Inguva, Yilin Wang
YouTube UGC Dataset for Video Compression Research
2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)

Raw Y plane value

Extract raw luma values from YUV input by keeping only the first plane

```
siti(extract_y(mov, 10))

ans = 10×2

  118.0413         0
  117.2755   33.7012
  116.8590   33.7242
  116.2051   33.7549
  116.3793   33.5950
  116.1294   33.6699
  116.1012   33.4861
  115.5719   33.5076
  115.1068   33.2865
  115.5687   33.6239
```

RGB to Gray Conversion

Many libraries read video and implicitly convert to RGB. Use MATLAB provided `rgb2gray` to extract only luma values

```
siti(extract_gray(mov, 10))

ans = 10×2

  136.4577         0
  135.4784   38.9945
  134.9973   39.0115
  134.2885   39.0512
  134.4655   38.8632
  134.1691   38.9509
  134.1383   38.7420
  133.5438   38.7586
  133.0275   38.5108
  133.5366   38.8975
```

YUV Limited (TV) to Full (PC) Range Conversion

Rescale 16-235 to 0-255 using `(Y - 16) / (235-16) * 255`

```
siti(extract_y_full(mov, 10))

ans = 10×2

  137.4454         0
  136.5536   39.2411
  136.0688   39.2679
  135.3073   39.3036
  135.5102   39.1174
  135.2192   39.2047
  135.1863   38.9907
  134.5701   39.0157
  134.0285   38.7582
  134.5663   39.1511
```

Review

- Both TI and SI values are impacted if calculations are performed in limited vs full range
- Review of existing implementations will show several inconsistencies (as of early October 2020)
- Illustrate the need for both a clear definition and available test vectors to help implementations verify correctness

Review of Existing Open Source Implementations

C++ implementation, using OpenCV

- https://github.com/Telecommunication-Telemedia-Assessment/SITI
  - Inconsistent handling of limited vs full range depending if the input is YUV or Y4M
  - Raw YUV input uses raw pixel values of the Y plane, preserving limited range values
  - Other input formats are read in RGB by OpenCV and converted to gray, yielding 0-255 range
  - Reports TI at frame N as a difference to frame N+1 instead of N-1, no value for last frame
  - Computes Sobel using reflection, but discards the outer border to keep only N-2 x M-2 pixels, which is fine
  - However, perhaps inadvertently applies the same mask for TI calculations

Review of Existing Open Source Implementations

Alternate Python implementation, using sk-video

- https://github.com/Telecommunication-Telemedia-Assessment/SITI
  - Explicitly disclaims that it may return different values than the C++ implementation in the same repository
  - Performs Sobel using reflection due to default mode of scipy.ndimage.sobel, does not discard outer border
  - Computes TI against the previous frame, reports 0 for the first frame
  - Input is always converted to RGB upon read by sk-video and therefore always uses subsequent gray conversion (0-255)
  - No option to read raw YUV files without implicit RGB conversion
  - There appears to be a bug in the output of SI yield very different range of values (500+)

Review of Existing Open Source Implementations

Python implementation by Werner Robitza, using PyAV for reading, SciPy for image processing tasks

- https://github.com/slhck/siti
  - Calibrated against the earlier C++ implementation
  - Points out the problem with lack of test vectors leading to differences between implementations
  - Performs Sobel using reflection due to default mode of scipy.ndimage.sobel, does not discard outer border
  - Computes TI against the previous frame, reports 0 for the first frame
  - Prior to 1.1 input was always converted to GRAY (0-255)
  - Recently added option for raw YUV input and noted the same problem with difference in limited range https://github.com/slhck/siti/issues/4
  - Now properly handles YUV in limited or full range by converting to 0-255 if input is raw YUV in limited range

Desired Outcomes

In order to avoid implementation specific inconsistencies, a revised definition would ideally ensure

- stable output for the same sequence if converted between limited and full range
- handling of higher bit depth formats while always keeping output in 0-255 range
- consistent definition of temporal measurement on the first frame of the sequence
- availability of reference implementation and test vectors to verify correctness of implementations

Next Steps

- seek feedback from VQEG on aligning existing implementations
- ideally align behavior across existing C++ and Python implementations
- develop community test vectors to ensure consistency among open-source implementations
- propose amendment at the next ITU-T full meeting

Questions