VVCPY: A Framework for Data Analysis and Visualization for Versatile Video Coding Standard

Luís Eduardo Pereira Mendes
lepmendes@inf.ufrgs.br

Federal University of Rio Grande do Sul
Institute of Informatics

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Recent Works on VVC Standard

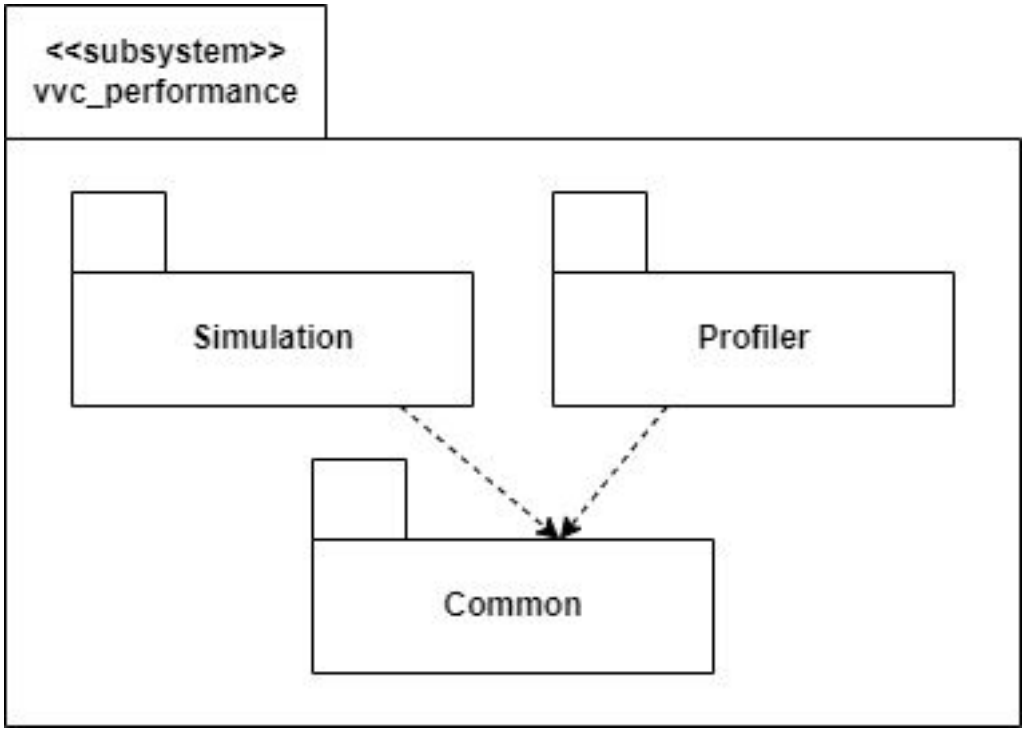
- Complexity reduction on **software level**;
- Implementation of **Hardware acceleration** on the system's hotspots;
- Use of **Approximate Computation** to increase performance without significantly losses.

Need of the Tool

- Abstraction of the process of **obtaining and manipulating data** from benchmarks;
- **Ease of development** of different test situations;
- **Reduced effort** to develop benchmarks.

Framework Design

- The software is organized as follows:



Framework Design - Data Structures

- Data structures inherited from **Pandas framework**

Frame	Bitrate	Y-PSNR	U-PSNR	V-PSNR	YUV	QP
f_{-1}	b_{-1}	y_{-1}	u_{-1}	v_{-1}	yuv_{-1}	qp_{-1}
f_0	b_0	y_0	u_0	v_0	yuv_0	qp_0
f_1	b_1	y_1	u_1	v_1	yuv_1	qp_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
f_n	b_n	y_n	u_n	v_n	yuv_n	qp_n

TABLE III: `VVC_output` structure

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Framework Design - Data Structures

Time(%)	cumulative time	self seconds	calls
t_0	ct_0	ss_0	c_0
t_1	ct_1	ss_1	c_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_n	ct_n	ss_n	c_n

TABLE IV: `Gprof_output` structure

self sec per call	total sec per call	name	class
ssc_0	tsc_0	f_0	cl_0
ssc_1	tsc_1	f_1	cl_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ssc_n	tsc_n	f_n	cl_n

TABLE V: `Gprof_output` structure (continuation)

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Framework Design - Data Structures

Version	video	configuration	frame	bd-rate
			f_0	BDR_0
		c_0	f_1	BDR_1
v_0	vi_0		$\vdots$	$\vdots$
			f_{j-1}	BDR_{j-1}
			f_0	BDR_j
		c_1	f_1	BDR_{j+1}
			$\vdots$	$\vdots$
			f_{j-1}	BDR_{2j-1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_n	vi_m	c_i	f_j	BDR_k

TABLE VI: `BD_Rate` structure

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Framework Design - Standard File Organization

- Output files have a standard name format, which identifies:
  - Version of the VTM;
  - Video File encoded;
  - Quantization parameter;
  - Encoder configuration (“AI”, “RA”, “LB”).

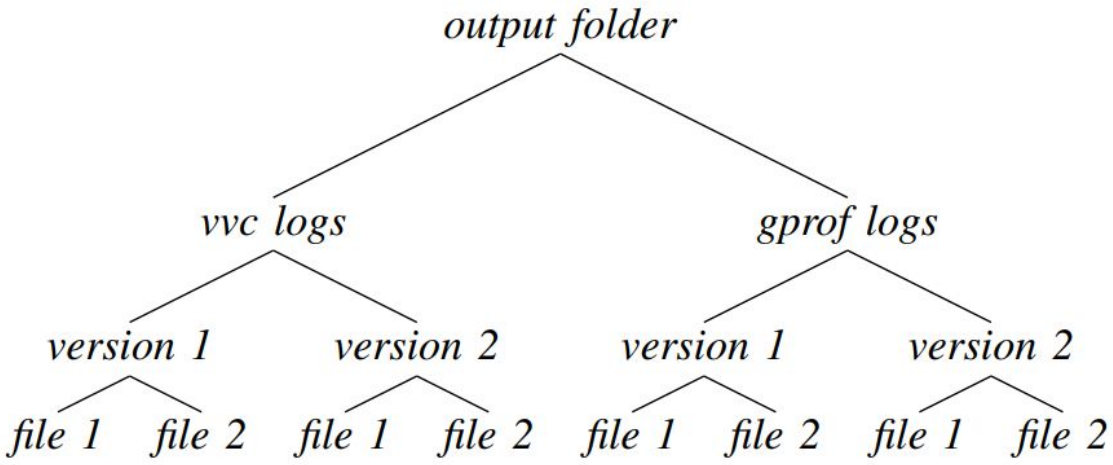


Fig. 1: Framework storage structure

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Framework Design - Standard File Organization

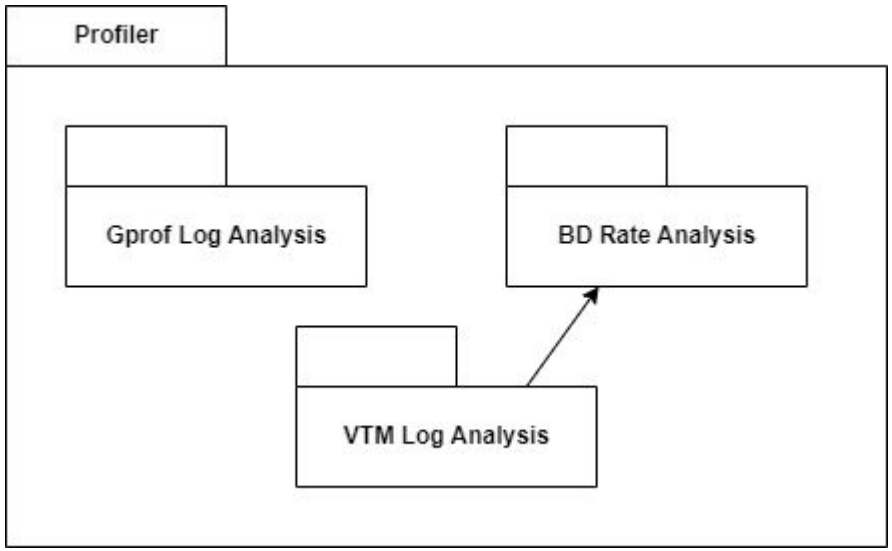
- VVCPY provides a set of methods to use the file organization to improve the development
- The methods consists in:
  - Get **video identifier**
  - Get path to **VTM Log**
  - Get path to **GProf Log**
  - Get path to **bin files**

Framework Design - Data Obtaining Process

- The Data Obtaining process is given by the **Simulation** part of the software
- Consists in:
  - Execute VTM
  - Create Simulation
- The Simulation created is composed by:
  - A set of ***videos*** to be encoded
  - A set of ***QPs***
  - A set of ***Encoder Configurations***
  - The ***Number of Frames***
  - A name of the ***Version*** (just for comparison purposes)

Framework Design - Data Processing

- The data processing step is divided into 3 different analysis:
  - **VTM Log** Analysis
  - **GProf Log** Analysis
  - **BD Rate** Analysis



Framework Design - Data Processing (GProf)

- The GNU Profiler Log Analysis is used to **find hotspots** in the VTM Software
- In the analysis it's possible to:
  - Read the GProf Log
  - Filter the information by
    - Class
    - Functions

Framework Design - Data Processing (VTM)

- The VTM Log is widely used to analyse information about the encoding process
- With the VVCPY it's possible to **read and process** these files
- It's used as input to the **BD Rate calculations**
- In this analysis process, it's possible to:
  - Read one or more **VTM Log Files** and compare the results
  - Include **custom parsing** methods using **regex**

Framework Design - Data Processing (BD Rate)

- The **BD Rate** is one of the most used metrics to video processing analysis
- It's calculated from the log obtained from VTM
- In this process, it's possible to:
  - Calculate **BD Rate** to a specific set of logs
  - Calculate **BD Rate** to a described set of logs, composed by:
    - A set of **Videos**
    - A set of **Versions**
    - A set of **QPs**
    - A set of **Encoding Configurations**

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Framework Design - Data Visualization

- The data visualization process is composed by a set of methods applied to the previous data structures
- It's possible to:
  - Plot graph of **PSNR by Bitrate**
  - Plot bar graph of **BD Rate**
  - Plot horizontal bar graph of the most used **methods** on VTM execution
  - Plot horizontal bar graph of the most **classes** on VTM execution

Usage Examples - Generating Benchmarks

- Using the `Simulation` methods

```
import vvcpy as vp

vtm_dir = '/home/VVCSoftware_VTM/'
cfg_dir = '/home/cfg-files'
out_dir = '/home/output'

encoder = ['RA', 'AI']
qps = [22, 27, 32, 37]

sim = vp.Simulation(
    n_frames=32,
    qps=qps,
    encoder=encoder
    )
sim.set_paths(out_dir, vtm_dir, cfg_dir)

sim.run_exec()
```

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Generating Benchmarks

- Result of `Simulation.run_exec()`:

Video Name	Encoder Configuration	Quantization Parameter
RaceHorses	RA	22
		27
		32
		37
	AI	22
		27
		32
		37

TABLE VII: VTM executions generated

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Making Changes on VTM

```
import vvcpy as vp

new_version = 'RdCostModif'
dst_file = '/home/VVCSoftware_VTM/source/Lib
    /CommmonLib/RdCost.cpp'
src_file = '/home/RdCostModif.cpp'

sim = vp.sim.Simulation()
sim.read_yaml('setup.yaml')
sim.change_version(
    new_version=new_version,
    old_file=dst_file,
    new_file=src_file
)
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage examples - Path generating

```
import vvcpy as vp

output_path = 'home/output/'
cfg         = 'AI'
file        = 'BasketballPass'
version     = 'Precise'
qp          = 22

path_to_vtm_log =
    vp.common.log_path.vvc_log_path(
    path = output_path,
    cfg = cfg,
    file = file,
    version = version,
    qps = qp,
    multiqp=False
)
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage examples - Path generating

```
qps = [22, 27, 32, 37]

paths_to_vtm_log =
    vp.common.log_path.vvc_log_path(
   path = output_path,
   cfg = cfg,
   file = file,
   version = version,
   qps = qps,
   multiqp=True
)
```

```
paths_to_gprof_log =
    vp.common.log_path.gprof_log_path(
   path = output_path,
   cfg = cfg,
   file = file,
   version = version,
   qps = qp,
   multiqp=False
)
```

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Obtaining Profiler Data

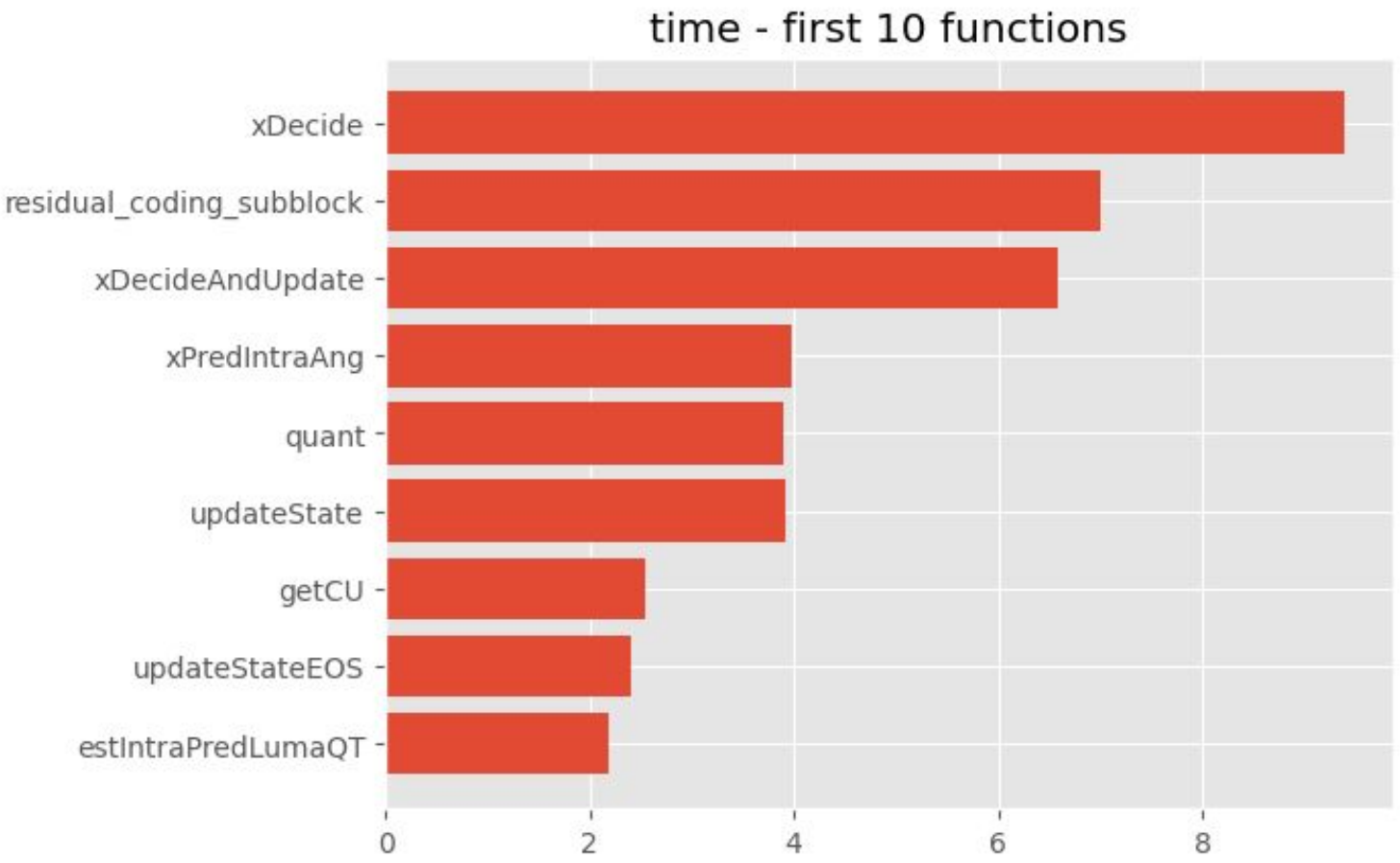
```
dfg = vp.prof.GprofDF()
dfg.read_file(path_to_gprof_log)
dfg.gplot()
dfg.to_csv('data2.csv')
```

	time	cum_sec	self_sec	calls	self_s_call	tot_s_call	namespace	class	function
0	9.4	160.43	160.43	2236926213.0	0.0	0.0	DQIntern	DepQuant	xDecide
1	7.0	279.86	119.43	181495104.0	0.0	0.0		CABACWriter	residual_coding_subblock
2	6.58	392.09	112.23	2236926213.0	0.0	0.0	DQIntern	DepQuant	xDecideAndUpdate
3	3.97	459.76	67.67	216562400.0	0.0	0.0		IntraPrediction	xPredIntraAng
4	3.91	526.44	66.68	2516527264.0	0.0	0.0	DQIntern	State	updateState
5	3.9	593.01	66.57	111658570.0	0.0	0.0	DQIntern	DepQuant	quant
6	3.63	654.94	61.93	2343283204.0	0.0	0.0	DQIntern	State	updateState
7	2.55	698.41	43.47	1945056926.0	0.0	0.0		CodingStructure	getCU
8	2.41	739.46	41.05	329577960.0	0.0	0.0	DQIntern	State	updateStateEOS
9	2.19	776.77	37.31	10647262.0	0.0	0.0		IntraSearch	estIntraPredLumaQT
10	1.84	808.24	31.47	11030197828.0	0.0	0.0			

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Finding path to a given file

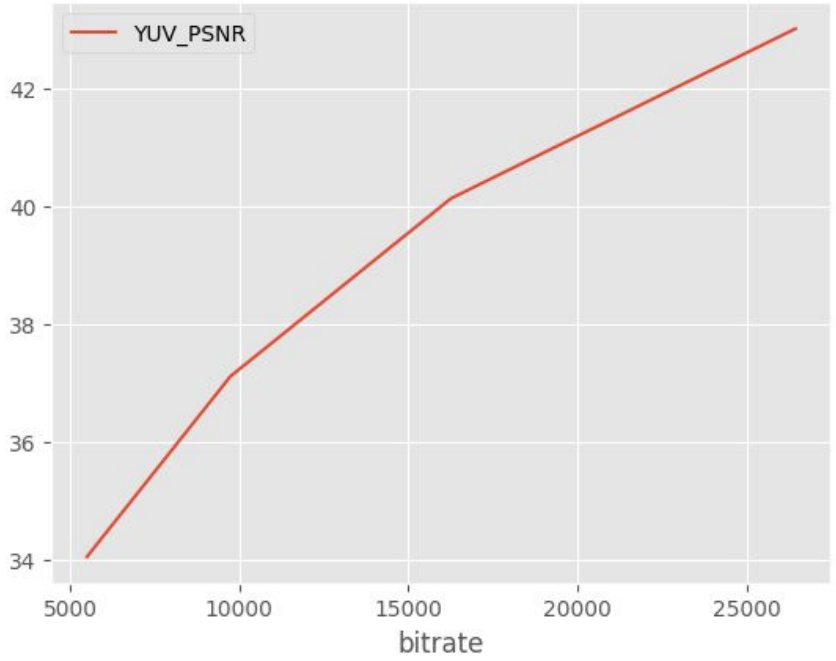


UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Obtaining VTM Log Data

```
dfv1 = vp.prof.VVC_Output()
dfv1 = dfv1.read_multifile(path_to_vtm_logs,
    qps)
dfv1.plot('bitrate', 'YUV_PSNR')
dfv1
```



	frame	t_frame	qp_offset	bitrate	Y_PSNR	U_PSNR	V_PSNR	YUV_PSNR	qp
0	-1			26428.74	42.3527	43.9969	45.5974	43.0089	22
0	-1			16261.2	39.3985	41.5764	42.7369	40.1305	27
0	-1			9742.14	36.1866	39.5352	40.2987	37.1102	32
0	-1			5508.12	32.853	38.0392	38.8779	34.0517	37

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf UFRGS

Usage Examples - Obtaining BD-Rate Values

```
dfv2 = vp.prof.VVC_Output()
dfv2 =
    dfv2.read_multifile(path_to_vtm_logs_2,
    qps)
```

```
dfb = dfb.calc_bdbr(dfv2, dfv1)
dfb.to_csv('data3.csv')
```

version	video	cfg	frame	bd-rate
Precise	BQMall	AI	-1	0.02683354950359096

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Download and Installation

- Open terminal and write down this command, making sure that git is installed:

```
git clone https://github.com/luiseduardomendes/vvc_performance.git
```

- then enter in the directory using:

```
cd vvc_performance
```

- and finally enter:

```
python3 -m pip install .
```

- After this the package was installed and can be used.

Institute of Informatics

Thank you!

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS