Contents

1. Introduction 1
1.1 Motivation 1
1.2 Problem Definition 1
1.3 Project Objective 2
1.4 The used tools in the project 3
2. System Analysis 4
2.1. Project Specifications 4
2.1.1 General Form of Programs 4
2.1.2 Formal Grammar 5
2.1.3 Parse Tree Sample 6
3. Constructs, Global Options, and Actions 8
4. System Design 19
4.1 System Architecture 19
Figure (10) 19
4.2 Syntax diagrams 19
5. Implementation and Testing 33
5.1 Implementation Details 33
5.2 Testing 34

1. Introduction

1.1 Motivation

Media production is a professional and hobbyist practice that involves splicing, editing, and processing collections of video, image, or sound files. It is a rapidly growing field with an increasing demand for efficient, high-quality tools.

Tools in the form of computer programs can generally be split into two types; graphical user interface (GUI), and script based. GUI programs involve interacting with visually displayed features, usually using the mouse to click on them. Whereas script based programs involve writing a script that gets executed by the machine into applying the program's features.

Working with GUI-based media editors can quickly become a repetitive and tiring task. Having a written description - a script - of the required edits be immediately implemented will increase productivity and make the entire editing process smoother.

1.2 Problem Definition

Design, implement, and test a domain specific scripting language with the purpose of manipulating media files. User-friendliness and general clarity of code is prioritized. The required external dependencies for installing the program should be minimized.

The script is to be interpreted into a sequence of system commands, which are then automatically executed in the correct and logical order.

1.3 Project Objective

Frame is an external domain specific language (DSL) for media editing and production, designed to be easily understood and written by both professional and hobby editors.

It is powered by the industry–level open source software project FFmpeg, a powerful and complex command line tool for encoding, decoding, editing, and various other manipulations of media.

Frame scripts are interpreted with the help of ANTLR, an advanced framework for parsing and code generation. Used extensively by Python, Objective-C, the parsing and querying of Twitter searches, and more. The process of developing Frame will be described in detail.

The project's codebase will consist almost entirely of C++ code, which is an ideal language for writing an interpreter, due to its speed and memory control. A free-flowing but structured coding style is favored, with any unnecessary abstraction kept to a minimum.

1.4 The used tools in the project

1.4.1 ANTLR

"ANTLR[1] (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build and walk parse trees." ~ Source: ANTLR homepage

ANTLR is responsible for generating the C++ code for our customized lexer and parser, based on the grammar that we will give it. The grammar for Frame can be seen in the "Project Specifications" section of this document.

ANTLR, however, is not a dependency for installing the final product, as it only generates sections of the code. The code is then compiled into an executable. This executable is what is included in the installation of the final product.

1.4.2 FFmpeg

"FFmpeg[2] is a free and open-source software project consisting of a suite of libraries and programs for handling video, audio, and other multimedia files and streams. At its core is the command-line ffmpeg tool itself, designed for processing of video and audio files."
~ Source: FFmpeg - Wikipedia

All media handling in Frame is handled by FFmpeg. This includes loading the files, any actions performed on the files, and saving the results.

FFmpeg is a required dependency for using Frame. An up-to-date version of the tool will be shipped with the installation process of the final product. Adding FFmpeg to the user's operating system's environment variables is a requirement.

2. System Analysis

2.1. Project Specifications

2.1.1 General Form of Programs

A general form of any Frame script consists of global options (optional), followed by one or more constructs, followed by zero or more actions. Here is a generic program, to illustrate:

```
// This is a comment

/*
Options
{
        output :
        reencode :
        debug:
        ... :
}
*/

// One or more constructs...

Audio a1
{
        path : ""       // All constructs must start with a path specification,
//followed by zero or more properties
        codec :
        ... :
        ... :
}

Frame f1
{
        path : ""
        format:
        resolution:
        ... :
}
```

```
Video v1
{
        path : ""
        framerate :
        ... :
        ... :
}

Video v2
{
        path : ""
        ... :
        ... :
        ... :
}

// ...followed by zero or more actions

trim v1 from “00:01:05” to “00:03:00" as v3   // "as .." is optional in all
// actions. If unwritten the edit output overrides the original video
merge a1 with v2  as vm
trim vm from “00:00:53” to “00:01:10” as final
```

2.1.2 Formal Grammar

The formal grammar that describes Frame is a context-free LL grammar. An LL grammar is designed to be parsed by an LL parser, which parses the input from **left** to right, and produces a **leftmost** derivation of the matched sentence. It also operates in a top-down recursion.

Snippet:

```
grammar Frame;

file : program EOF
;

program : (COMMENT | NEWLINE)* construct (COMMENT | construct | action |
NEWLINE)*
```

```
;

construct : type=(AUDIO | FRAME | VIDEO) NAME (NEWLINE)? '{' NEWLINE path
NEWLINE (property)* '}' NEWLINE
;
NAME : ALPHA ( ALPHA | DIGIT )*
;

TIME : '"' DIGIT DIGIT ':' DIGIT DIGIT ':' DIGIT DIGIT '"'
;

DIGIT: [0-9]
;

ALPHA: [a-zA-Z_]
;

WS : [ \t]+ -> skip
;
```

2.1.3 Parse Tree Sample

To illustrate the specifications of parsing a Frame script, let's take a look at an example working .frame file, and the generated parse tree our grammar leads to.

"example.frame" contains the following script:

```
Video example
{
        path : "path\to\example_video.mp4"
}

trim example from "00:00:07" to "00:00:13" as example_trimmed
```

Figure (3) shows the generated parse tree hierarchy.

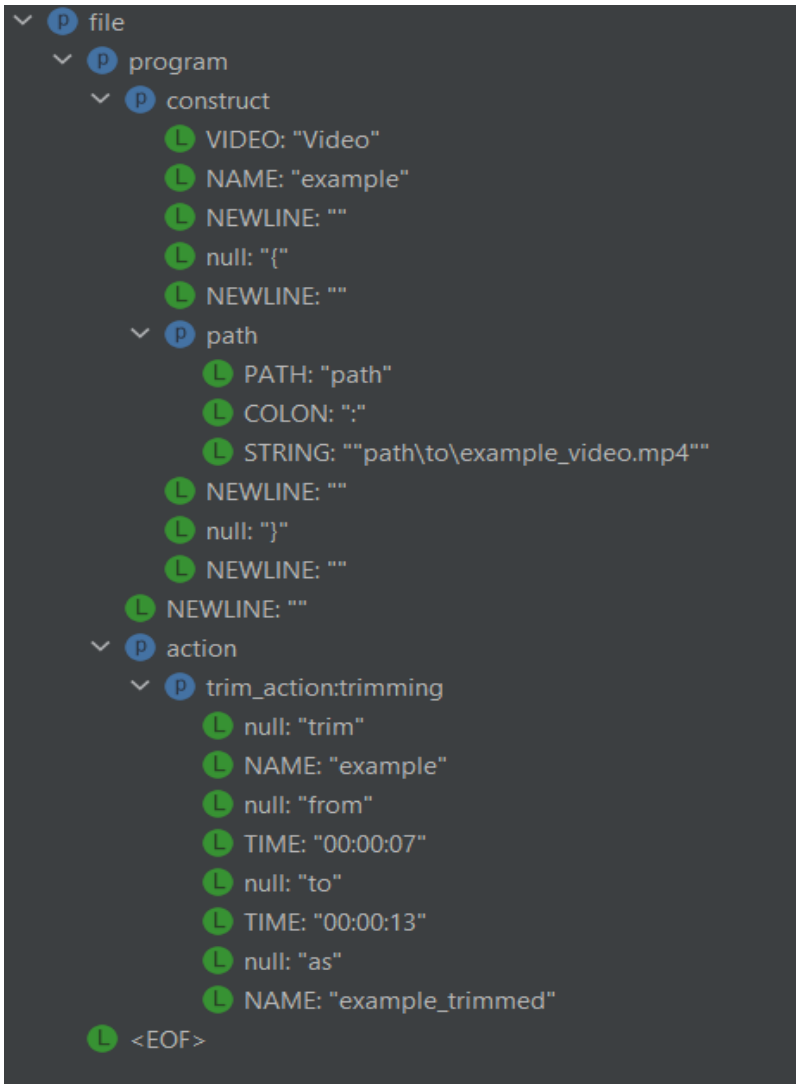


Figure (3): Node hierarchy of example.frame parse tree

3. Constructs, Global Options, and Actions

This section defines the available data types (declared as constructs), some of the proposed actions, and some of the available global options, within the Frame language.

Data Type (Construct)	Description	Properties
Audio	An audio file	path codec sampling_rate
Frame	A still image	path codec resolution
Video	A video file	path codec format resolution framerate

Table (1)

Global Option	Description	Values
output	Specifies a global output directory.	<path string>
reencode	Specifies whether to reencode a file after applying an action to it or not. Reencoding is slower but more accurate.	yes no
overwrite	Specifies whether to overwrite the original construct or not	yes no
debug	Specifies whether to enable debugging mode or not. Debugging mode is a mode that allows users to observe what happens during action execution to debug it.	yes no

Table (2)

Table (3)

Action	**Applies to**	**Description**	**Form**
trim	Video Audio	Trims the video or audio file	trim <name> from "<HH:MM:SS>" to "<HH:MM:SS>" as <name> trim <name> from "<HH:MM:SS>" to "<HH:MM:SS>" save to "full path"
Extract audio from video	Video	Extracts an audio file out of a video file	extract audio from <name> format <AUDIO_FORMAT> as <name> extract audio from <name> format <AUDIO_FORMAT> save to "full path"
Extract frames from video	Video	Extracts frames from a video file.	extract frames from <name> format <FRAME_FORMAT> as <name> extract frames from <name> format <FRAME_FORMAT> save to "full path" extract frames from <name> format <FRAME_FORMAT> ratio <integer number> <integer number> as <name> extract frames from <name> format <FRAME_FORMAT> ratio <int number> <int number> save to "full path"
Merge audio with video	Audio Video	Merges an audio file with a video file (Produces a video file whose new audio is the specified audio file)	merge <audio name> with <video name> as <name> merge <audio name> with <video name> save to "full path"
Concatenate	Video Audio Frame	Concatenates media files but they all have to be of the same type (audio, frame, video) and have	concatenate <name> <name> <name> <name> as <name> concatenate <name> <name>

		the same resolution in case they were frames or videos.	<name> <name> save to “full path”
overlay	Video Frame	Overlays video/frame on a video file	overlay <name> on <name> at position <int number> <int number> as <name> overlay <name> on <name> at position <int number> <int number> save to “full path”
crop	Video Frame	Crops a video/frame file	crop <name> width <int number> height <int number> at position <int number> <int number> as <name> crop <name> width <int number> height <int number> at position <int number> <int number> save to “full path” crop <name> width <int number> height <int number> as <name> crop <name> width <int number> height <int number> save to “full path”
rotate	Video Frame	Rotates video/frame files by a specific angle.	rotate <name> <int number> as <name> rotate <name> <int number> save to “full path”
flip	Video Frame	Flips a video/frame file horizontally/vertically.	flip <name> hflip as <name> flip <name> hflip save to “full path” flip <name> vflip as <name> flip <name> vflip save to “full path”
saturation	Video Frame	Changes the saturation of a video/frame by a specific amount.	set saturation for <name> <float number> as <name> set saturation for <name> <int

			number> as <name> set saturation for <name> <float number> save to “full path” set saturation for <name> <int number> save to “full path”
gamma	Video Frame	Changes gamma value of a video/frame by a specific amount.	set gamma for <name> <float number> as <name> set gamma for <name> <int number> as <name> set gamma for <name> <float number> save to “full path” set gamma for <name> <int number> save to “full path”
brightness	Video Frame	Sets the brightness level of a video/frame to a specific value.	set brightness for <name> -<float number> as <name> set brightness for <name> -<int number> as <name> set brightness for <name> -<int number> save to “full path” set brightness for <name> -<float number> save to “full path”
contrast	Video Frame	Sets the contrast level of a video/frame to a specific value.	set contrast for <name> -<float number> as <name> set contrast for <name> -<int number> as <name> set contrast for <name> -<int number> save to “full path” set contrast for <name> -<float number> save to “full path”
framerate	Video	Changes the framerate of a video file to a specific value.	set framerate for <name> <int number> as <name>

			set framerate for <name> <int number> save to “full path”
volume	Video Audio	Sets the volume of a video/audio file to a specific value.	set volume for <name> <int number> as <name> set volume for <name> <int number> save to “full path”
scale	Video Frame	Scales/Resizes a video/frame file according to the specified width and height values.	scale <name> width <int number> height <int number> as <name> scale <name> width <int number> height <int number> save to “full path”
Extract N subtitles from a video	Video	Extracts a previously specified number of subtitles from a video file as SRT files.	extract <int number> subtitles from <name> save to “full path”
Extract Kth subtitle from a video	Video	Extracts subtitle number ‘K’ from a video file. Example: if a video has 5 subtitles, we can extract the 2nd subtitle (subtitle number 2).	extract subtitle number <int number> from <name> save to “full path”
Add subtitles to a video	Video	Adds a selectable subtitle (SRT) to a video file (The subtitle Can be selected from options in a video player).	add subtitles “full path” for <name> as <name> add subtitles “full path” for <name> save to “full path”
sharpen	Video Frame	Sharpens a video/frame file by a specific value.	sharpen <name> width <int number> height <int number> intensity <int number> as <name> sharpen <name> width <int number> height <int number> intensity <int number> as save to “full path” sharpen <name> width <int number> height <int number> intensity <float number> as

			<name> sharpen <name> width <int number> height <int number> intensity <float number> save to “full path” sharpen <name> width <int number> height <int number> as <name> sharpen <name> width <int number> height <int number> as <name> sharpen <name> width <int number> height <int number> save to “full path” sharpen <name> intensity <int number> as <name> sharpen <name> intensity <int number> save to “full path” sharpen <name> intensity <float number> as <name> sharpen <name> intensity <float number> save to “full path”
blur	Video Frame	Blurs a video/frame file by a specific value. (The blur intensity is optional)	blur <name> width <int number> height <int number> intensity <int number> as <name> blur <name> width <int number> height <int number> intensity <int number> as save to “full path” blur <name> width <int number> height <int number> intensity <float number> as <name> blur <name> width <int number> height <int number> intensity

			<float number> save to “full path” blur <name> width <int number> height <int number> as <name> blur <name> width <int number> height <int number> as <name> blur <name> width <int number> height <int number> save to “full path” blur <name> intensity <int number> as <name> blur <name> intensity <int number> save to “full path” blur <name> intensity <float number> as <name>
convert	Video Frame Audio	Converts video/frame/audio’s format to a different format.	convert <name> format <format> as <name> convert <name> format <format> save to “full path”
Noise reduction	Video Audio	Reduces the audio noise of an audio file or a video file. (Mix value is optional, it specifies how visible the noise is)	reduce noise for <name> as <name> reduce noise for <name> save to “full path” reduce noise for <name> mix <float number> as <name> reduce noise for <name> mix <int number> as <name> reduce noise for <name> mix <float number> save to “full path”
Normalization	Video Audio	Normalizes the audio of an audio file or a video file.	normalize <name> loudness <float or int number> range <float or int number> peak <float

		The type of normalization used is loudness normalization.	or int number> as <name> normalize <name> loudness <float or int number> range <float or int number> peak <float or int number> save to “full path”
bass	Video Audio	Changes an audio’s bass for an audio file or a video file.	set bass for <name> gain -<float or int number> frequency <float or int number> as <name> set bass for <name> gain -<float or int number> frequency <float or int number> save to “full path”
treble	Video Audio	Changes an audio’s treble for an audio file or a video file.	set treble for <name> gain -<float or int number> frequency <float or int number> as <name> set treble for <name> gain -<float or int number> frequency <float or int number> save to “full path”
bitscope	Video Audio	Visualizes the audio of a video file or an audio file into a bitscope.	show bitscope for <name> colors <color> <color> width <int number> height <int number> as <name> show bitscope for <name> colors <color> <color> width <int number> height <int number> save to “full path”
histogram	Video Audio	Visualizes the audio of a video file or an audio file into a histogram. (dmode values can be ‘single’ or ‘separate’ - without single quotations.)	show audio histogram for <name> display mode <dmode> as <name> show audio histogram for <name> display mode <dmode> save to “full path”
phasemeter	Video Audio	Visualizes the audio of a video file or an audio file into a phasemeter.	show phasemeter for <name> as <name> show phasemeter for <name> save to “full path”

vectorscope	Video Audio	Visualizes the audio of a video file or an audio file into a vectorscope.	show audio vectorscope for <name> drawing mode <drawingMode> as <name>
CQT	Video Audio	Visualizes the audio of a video file or an audio file into a CQT (Constant-Q-transform).	show cqt for <name> as <name> show cqt for <name> save to “full path”
frequencies	Video Audio	Visualizes the audio of a video file or an audio file into an audio power spectrum. **Display mode** is how an audio power spectrum is displayed. **Values**: line, bar, dot. **Channel mode** is whether each audio channel is displayed separately or not - in the same audio spectrum file. **Values**: combined, separate.	show frequencies for <name> display mode <displayMode> colors <color> <color> channel mode <channelMode> as <name> show frequencies for <name> display mode <displayMode> colors <color> <color> channel mode <channelMode> save to “full path”
spatial	Video Audio	Visualizes the relationship between two audio channels of an audio file or a video file.	show spatial for <name> as <name> show spatial for <name> save to “full path”
spectrum	Video Audio	Visualizes the audio of a video file or an audio file into an audio frequency spectrum. **Display mode** is how audio frequency spectrum channels are displayed. **Values**: combined, separate	show spectrum for <name> as <name> show spectrum for <name> save to “full path” show spectrum for <name> display mode <displayMode> as name
Show volume	Video	Creates a video output	show volume for <name> as

	Audio	showing the volume of the audio provided.	<name> show volume for <name> save to “full path”
super2xsai	Video Frame	Scales a frame/video by a factor of 2 (2x) using a pixel art scaling algorithm. (Preferably used with 2D sprites)	scale pixel art <name> as <name> scale pixel art <name> save to “full path”
sobel	Video Frame	Applies sobel filter (also called sobel operator) on video/frame files	sobel <name> as <name> sobel <name> save to “full path” sobel intensity <float or int number> as <name>
mix	Video	Mixes successive video frames. The number of successive frames to be mixed can be specified (Optional)	mix frames <int number> for <name> as <name> mix frames <int number> for <name> save to “full path” mix frames for <name> as <name>
Embedding subtitles	Video	Embeds/Burns a subtitle file into a video making them inseparable.	embed subtitles “full path” in <name> as <name> embed subtitles “full path” in <name> save to “full path”

Compatible video codecs

Format Codecs	MP4	OGG	WebM
VP8	No	Yes	Yes
VP9	Yes	Yes	Yes
libvps	No	Yes	Yes
H264	Yes	No	No
H265	Yes	No	No

Table (4)

Compatible audio codecs

Format Codecs	MP4	OGG	WebM
FLAC	Yes	Yes	No
OPUS	Yes	Yes	Yes
MP3	Yes	No	No
AAC	Yes	No	No

Table (5)

4. System Design

4.1 System Architecture

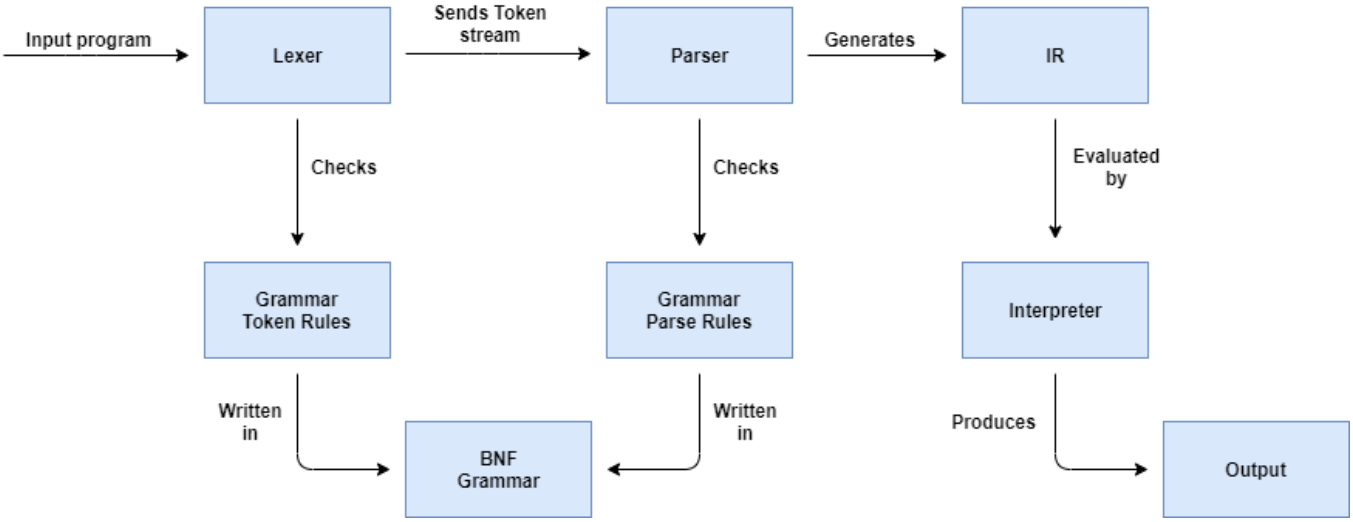
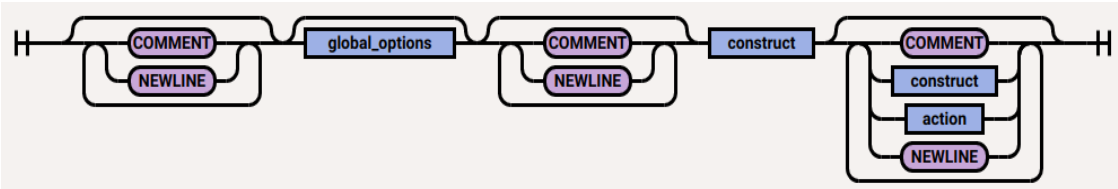
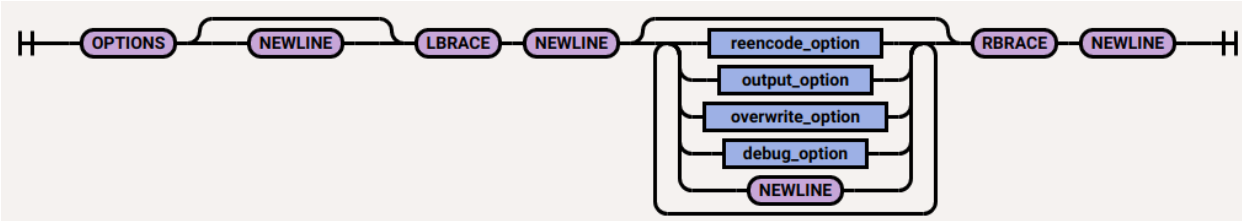
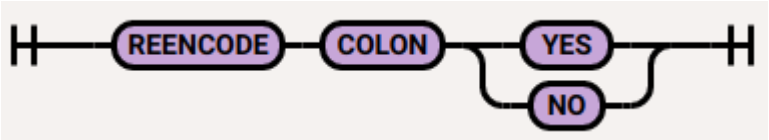
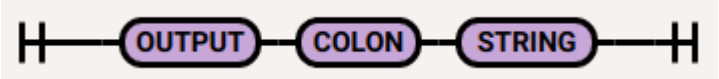
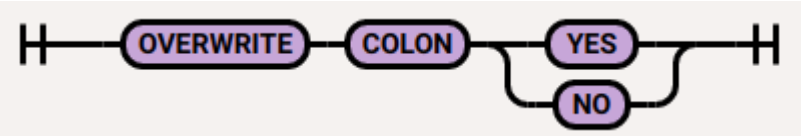
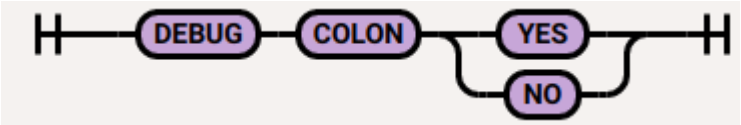
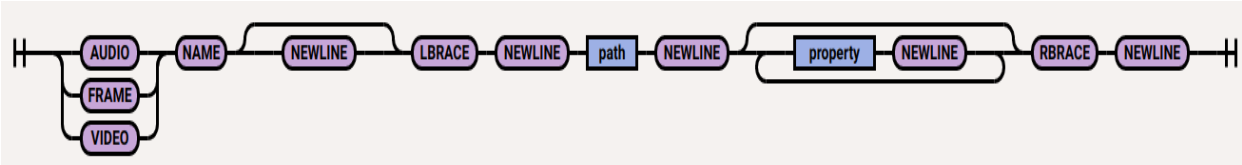
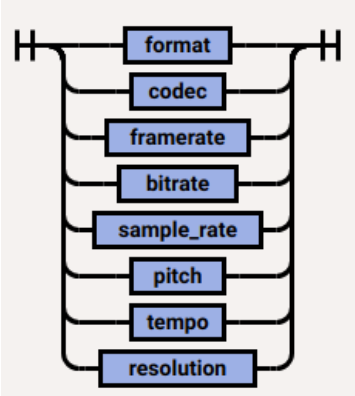
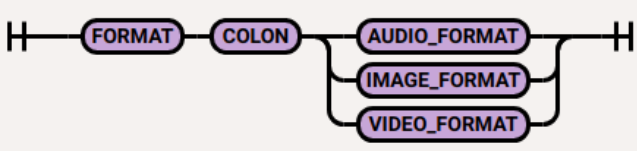
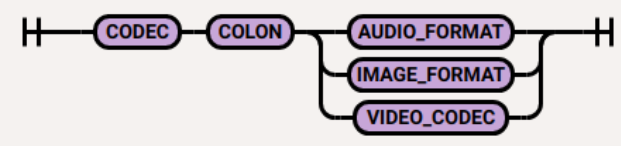
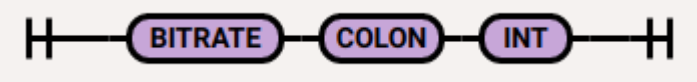


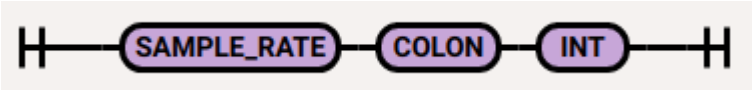
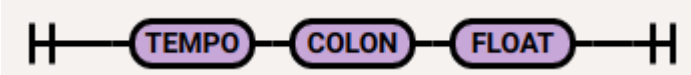
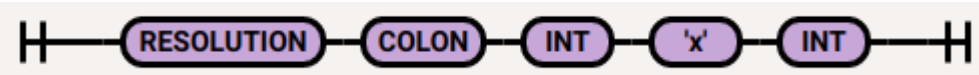
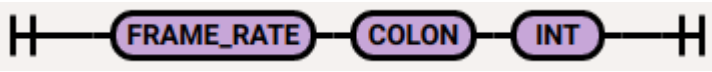
Figure (10)

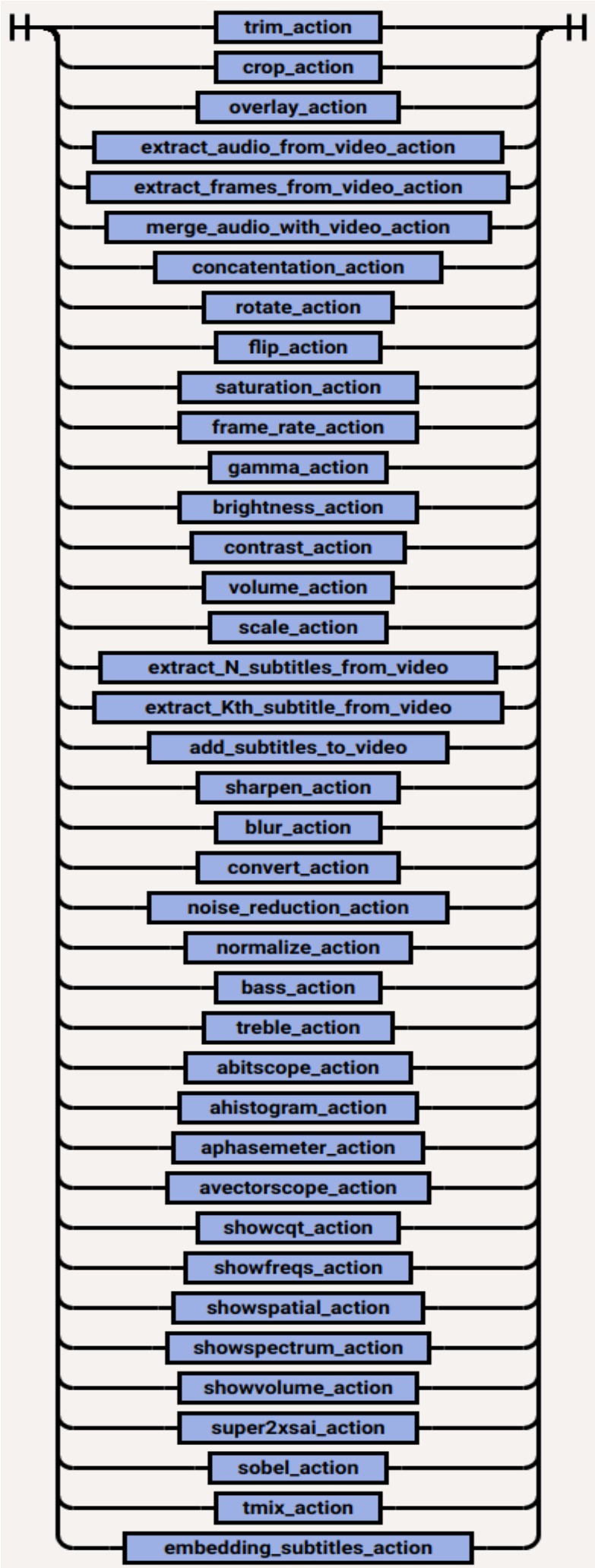
4.2 Syntax diagrams

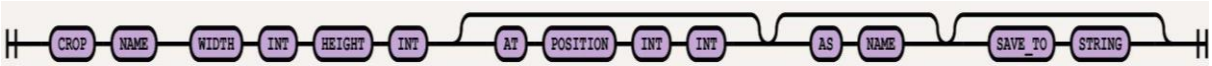
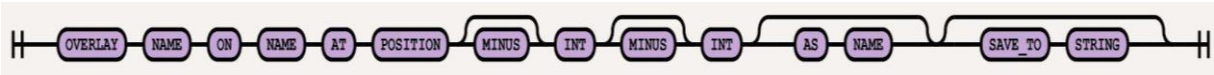
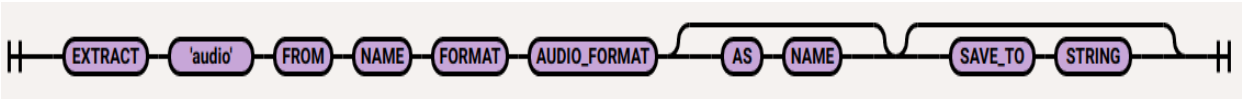
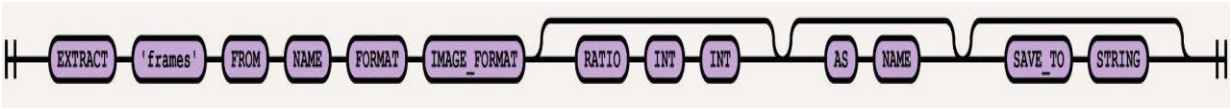
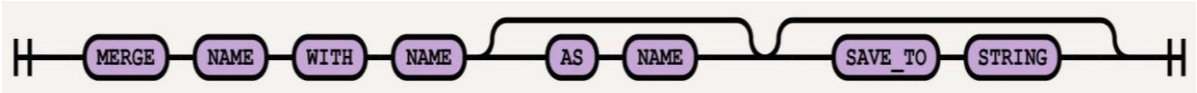
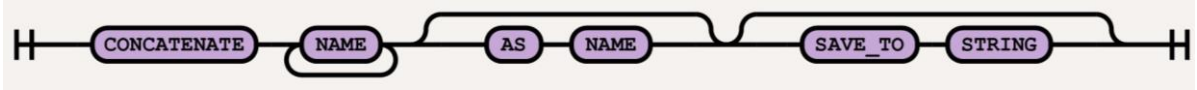
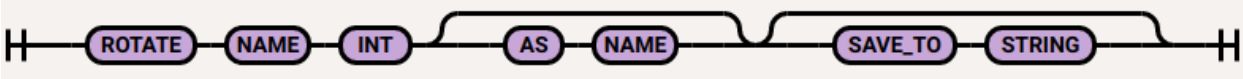
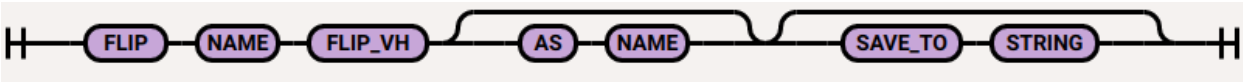
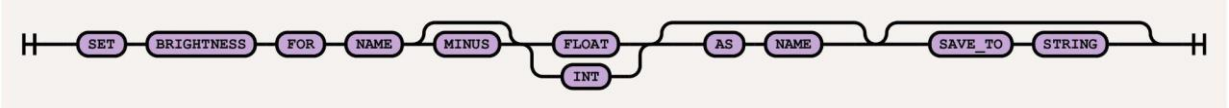
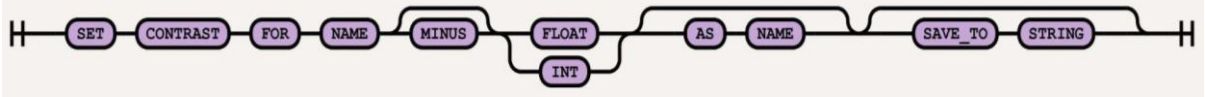
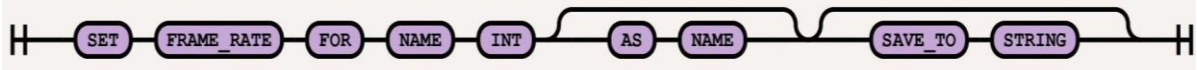
Table (7)

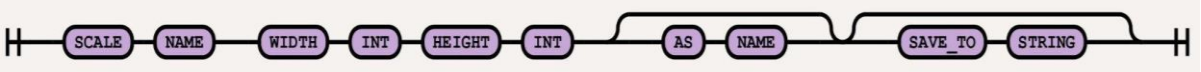
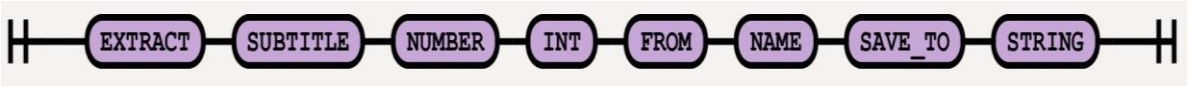
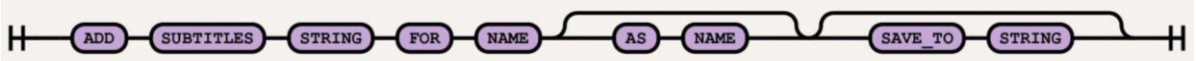
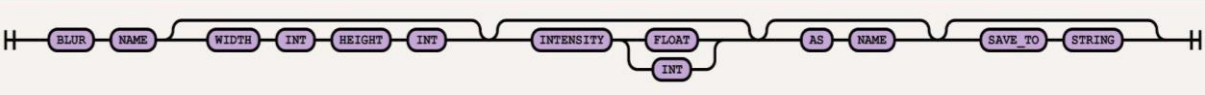
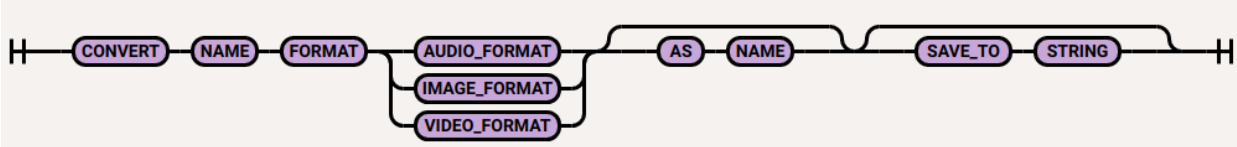
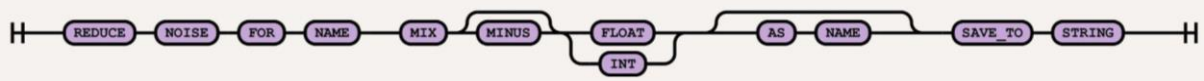
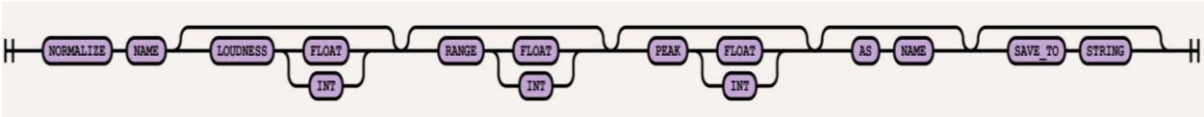
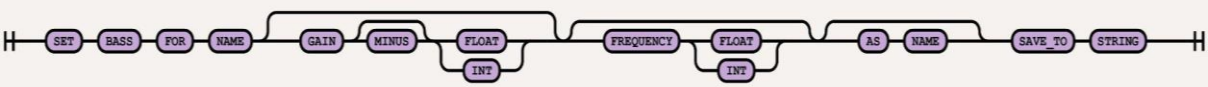
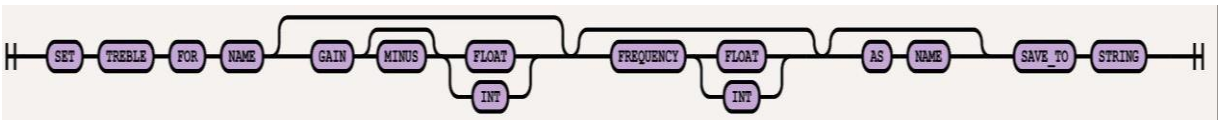
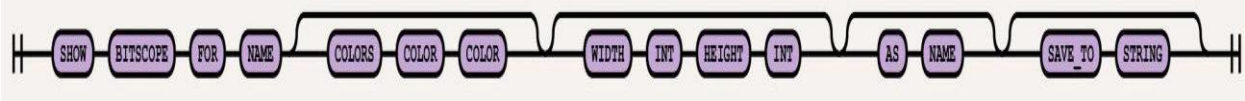
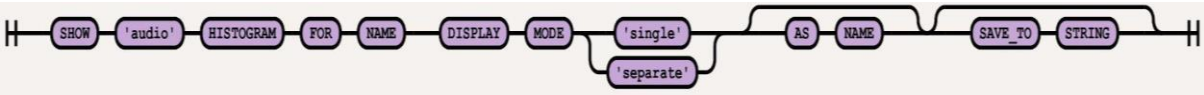
Rule/ Token	Diagram
file	
program	
global_options	

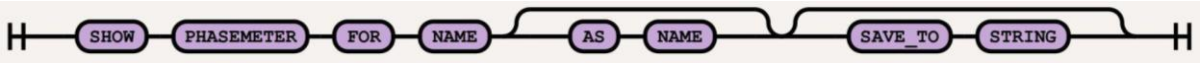
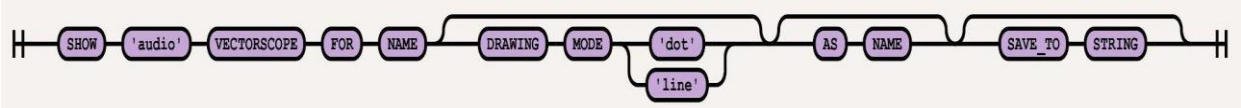
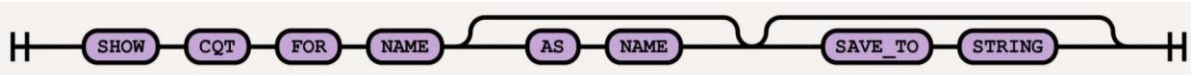
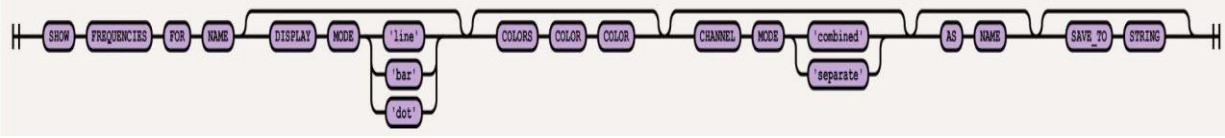
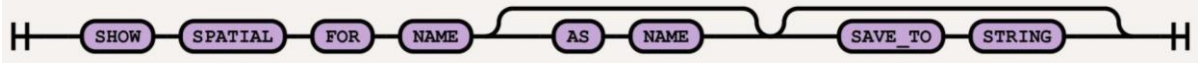
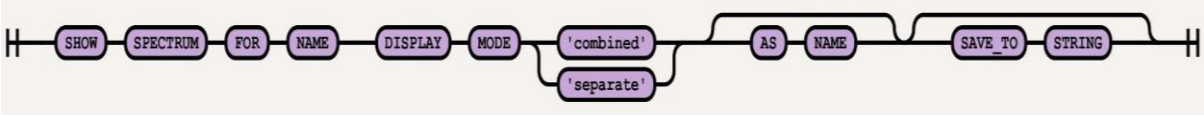
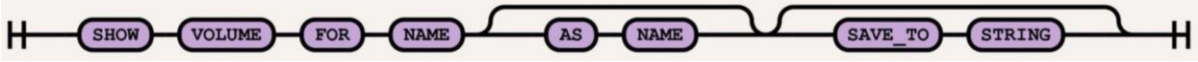
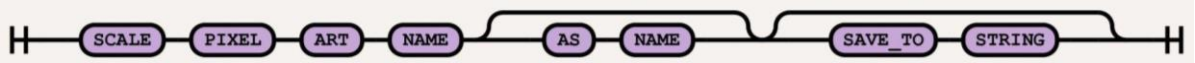
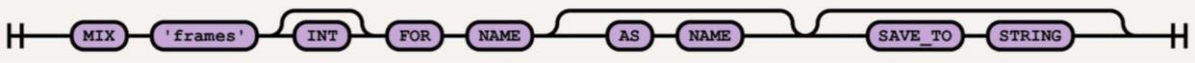
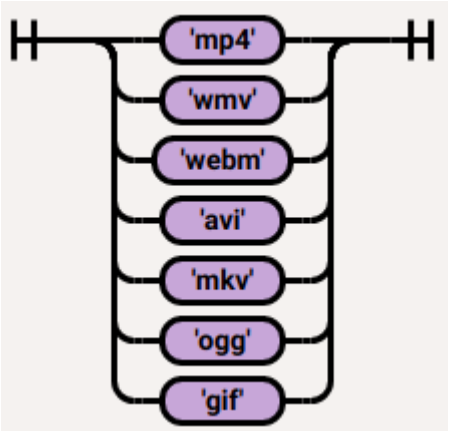
reencode_option	
output_option	
overwrite_option	
debug_option	
construct	
path	
property	
format	
codec	
bitrate	

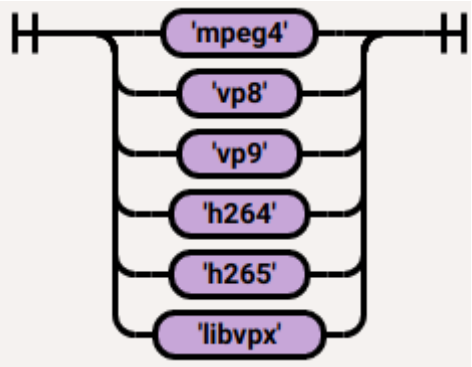
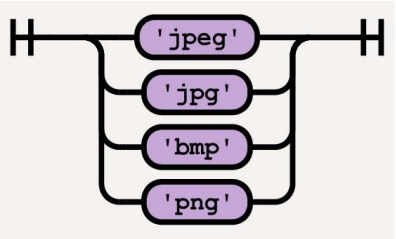
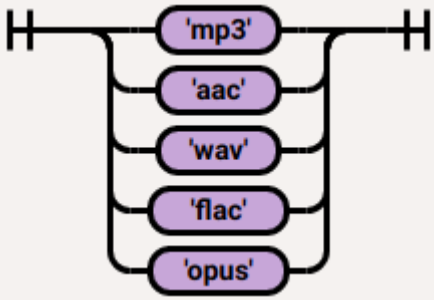
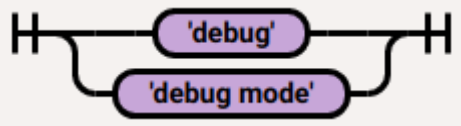
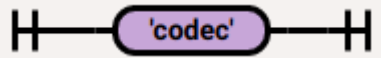
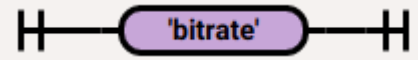
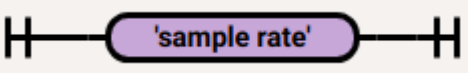
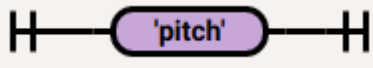
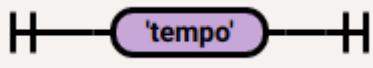
sample_rate	
pitch	
tempo	
resolution	
framerate	

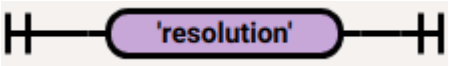
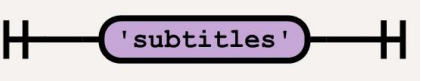
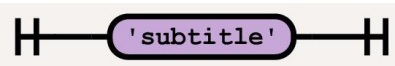
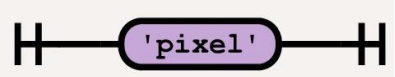
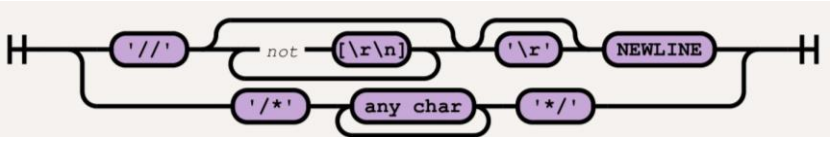
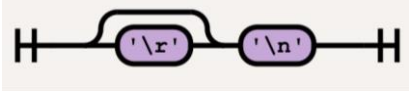
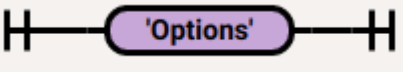
action	 trim_action crop_action overlay_action extract_audio_from_video_action extract_frames_from_video_action merge_audio_with_video_action concatentation_action rotate_action flip_action saturation_action frame_rate_action gamma_action brightness_action contrast_action volume_action scale_action extract_N_subtitles_from_video extract_Kth_subtitle_from_video add_subtitles_to_video sharpen_action blur_action convert_action noise_reduction_action normalize_action bass_action treble_action abitscope_action ahistogram_action aphasemeter_action avectorscope_action showcqt_action showfreqs_action showspatial_action showspectrum_action showvolume_action super2xsai_action sobel_action tmix_action embedding_subtitles_action

trim_action	
crop_action	
overlay_action	
extract_audio_from_video_action	
extract_frames_from_video_action	
merge_audio_with_video_action	
concatentation_action	
rotate_action	
flip_action	
saturation_action	
gamma_action	
brightness_action	
contrast_action	
frame_rate_action	

volume_action	
scale_action	
extract_N_subtitles_from_video	
extract_Kth_subtitles_from_video	
add_subtitles_to_video	
sharpen_action	
blur_action	
convert_action	
noise_reduction_action	
normalize_action	
bass_action	
treble_action	
abitscope_action	
ahistogram_action	

aphasemeter_action	
avectorscope_action	
showcqt_action	
showfreqs_action	
showspatial_action	
showspectrum_action	
showvolume_action	
super2xsai_action	
sobel_action	
tmix_action	
embedding_subtitles_action	
VIDEO_FORMAT	

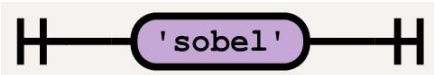
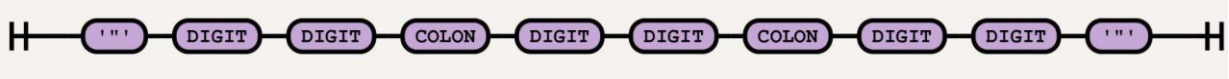
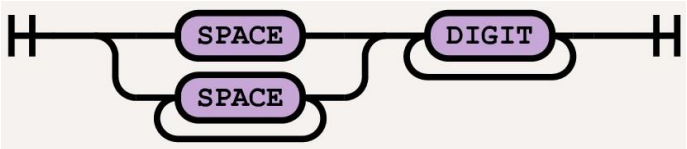
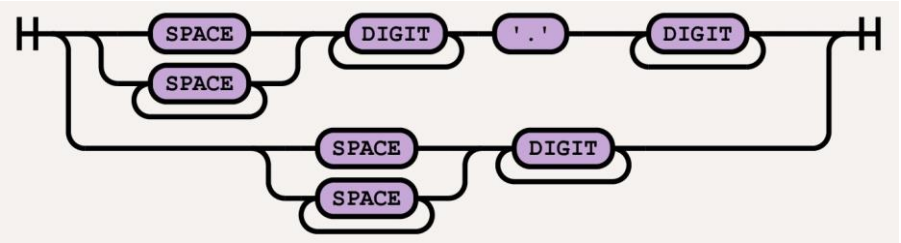
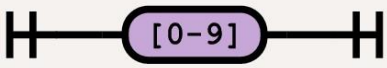
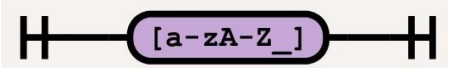
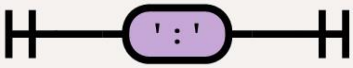
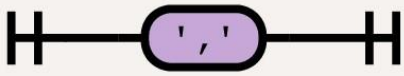
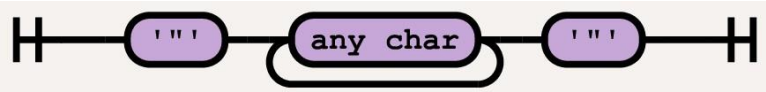
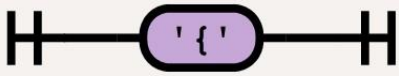
VIDEO_CODEC	
IMAGE_FORMAT	
AUDIO_FORMAT	
DEBUG	
CODEC	
BITRATE	
SAMPLE_RATE	
PITCH	
TEMPO	

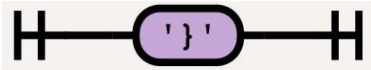
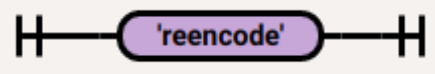
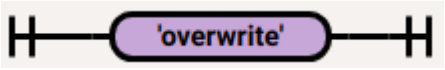
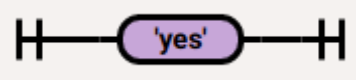
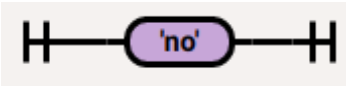
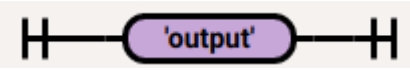
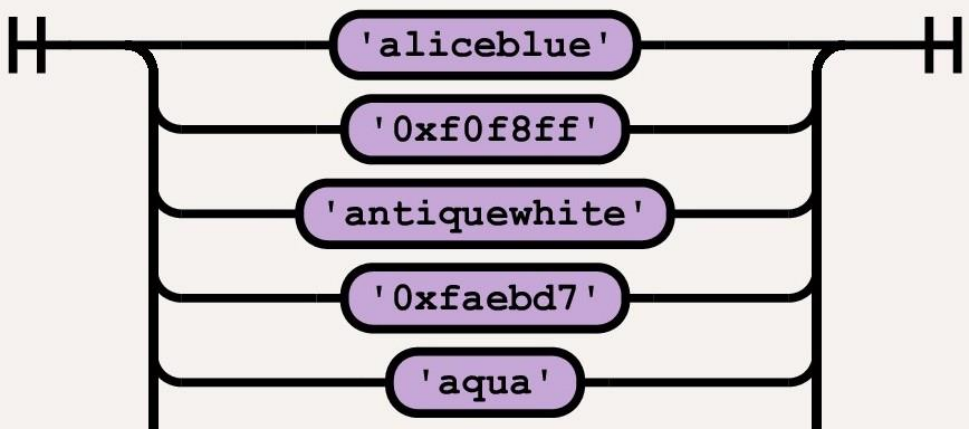
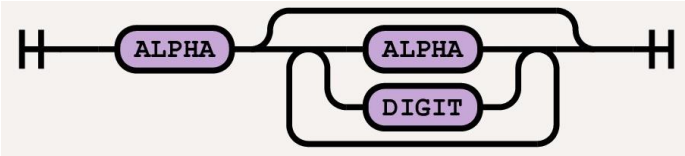
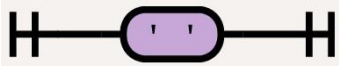
RESOLUTION	
SAVE_TO	
NUMBER	
SUBTITLES	
SUBTITLE	
FLIP_VH	
PIXEL	
ART	
FORMAT	
COMMENT	
NEWLINE	
OPTIONS	
AUDIO	
FRAME	
VIDEO	
PATH	

TRIM	'trim'
CROP	'crop'
FRAME_RATE	'framerate'
EXTRACT	'extract'
RATIO	'ratio'
OVERLAY	'overlay'
MERGE	'merge'
POSITION	'position'
SATURATION	'saturation'
GAMMA	'gamma'
CONTRAST	'contrast'
ROTATE	'rotate'
FLIP	'flip'
BRIGHTNESS	'brightness'
CONCATENATE	'concatenate'
SCALE	'scale'
SET	'set'
VOLUME	'volume'

HEIGHT	'height'
WIDTH	'width'
SHARPEN	'sharpen'
BLUR	'blur'
CONVERT	'convert'
ADD	'add'
FROM	'from'
TO	'to'
AS	'as'
WITH	'with'
FOR	'for'
ON	'on'
AT	'at'
INTENSITY	'intensity'
REDUCE	'reduce'
NOISE	'noise'
MIX	'mix'
NORMALIZE	'normalize'

LOUDNESS	'loudness'
RANGE	'range'
PEAK	'peak'
TREBLE	'treble'
BASS	'bass'
GAIN	'gain'
FREQUENCY	'frequency'
SHOW	'show'
BITSCOPE	'bitscope'
HISTOGRAM	'histogram'
DISPLAY	'display'
MODE	'mode'
PHASEMETER	'phasemeter'
VECTORSCOPE	'vectorscope'
DRAWING	'drawing'
CQT	'cqt'
FREQUENCIES	'frequencies'
SPATIAL	'spatial'

SPECTRUM	
COLORS	
CHANNEL	
SOBEL	
EMBED	
IN	
TIME	
INT	
FLOAT	
DIGIT	
ALPHA	
COLON	
MINUS	
COMMA	
STRING	
LBRACE	

RBRACE	
REENCODE	
OVERWRITE	
YES	
NO	
OUTPUT	
COLOR	
NAME	
SPACE	
WS	

5. Implementation and Testing

5.1 Implementation Details

Writing an interpreted language involves several stages. Firstly, the input program is fed into the **lexer**. The lexer reads the entire script character by character, and produces a stream of identified words, known as **tokens**. The tokens are then fed into the **parser**, which matches the set of tokens to one, and only one, grammar rule. The parser then constructs an **intermediate representation (IR)**. The IR is generally a tree hierarchy of statements, usually either in the form of an **abstract syntax tree (AST)**, or a **parse tree**.

An AST abstracts away some syntactical elements of the source program, whereas a parse tree retains more information, making the parse tree more detailed. Our project deals with parse trees.

The tree is then evaluated sequentially using a form of depth-first search. Once all the required semantics of a node-statement are gathered and translated, the statement is executed. The final result of this process is the **output**.

Our program's output is a sequence of system commands, which are executed by the operating system. These commands rely on the tool FFmpeg.

The lexer and parser subprograms are generated by the ANTLR framework, in C++ (MSVC), using a formatted grammar file. The grammar that ANTLR accepts is heavily based on the Backus-Naur Form (BNF) grammar for language specification.

5.2 Testing

Test Scenario ID	Test Scenario Description
TS_Frame_00	Verify the Language Syntax

Test Case ID	Test Case Description	Test Steps	Preconditions	Test Data	Postconditions	Expected Result	Actual Result	Status

TC_Syntax_01	Accepted Syntax	1- Define valid construct
		2-run
			Syntax is Valid	Video v1
				{
				path: "E:\videos\video.mp4"
				format : avi
				codec: h264
				codec: mp3
				}
					No error message	Run Successfully and format and codecs are changed	v1: Converting video format, this may take a while...
							v1: Converting codec, this may take a while...
							v1: Converting codec, this may take a while...
								Passed

TC_Syntax_02	Wrong Syntax	1- Define Invalid construct
		2-run
			Syntax is Invalid	Farme f1
				{
				path:"E:\images\frame.png"
				}
					An Error Message appears to the User	Error mismatched input	line 1:0 mismatched input 'Farme' expecting {COMMENT, NEWLINE, 'Options', 'Audio', 'Frame', 'Video'}	Passed

TC_Syntax_03	Missing Syntax	1- Define global options With missing Parameter
		2- Define valid construct
		3- run
			Missing Parameter	Options
				{
				overwrite: yes
				reencode: yes
				output: "E:\"
				debug:
				}
				
				Frame f1
				{
				path: "E:\images\frame.png"
				}
					An Error Message appears to the User	Error missing 'yes' or 'no' with debug	line 6:11 missing {'yes', 'no'} at '\n'	Passed

```
E:\runnable>frame.exe test.frame
v1: Converting video format, this may take a while...
v1: Converting codec, this may take a while...
v1: Converting codec, this may take a while...
```

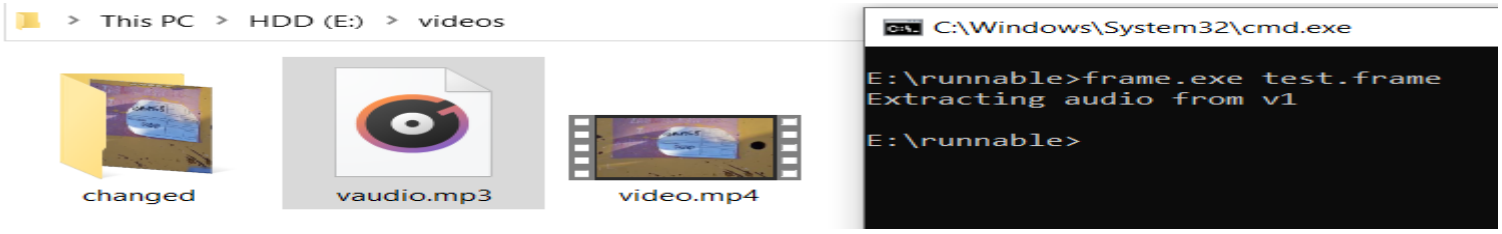
```
E:\runnable>frame.exe test.frame
line 1:0 mismatched input 'Farme' expecting {COMMENT, NEWLINE, 'Options', 'Audio', 'Frame', 'Video'}
```

```
E:\runnable>frame.exe test.frame
line 6:11 missing {'yes', 'no'} at '\n'
```

Test Scenario ID	**Test Scenario Description**
TS_Frame_01	Verify extract audio from video functionality

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

TC_extract_audio_01	Valid file type	1- define valid construct
		2- run
			Define valid (video) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				extract audio from v1 format mp3 as vaudio
					User should be able to find audio extracted from file_construct	file_construct extracted successfully	Extracting audio from f1	Passed

TC_extract_audio_02	Invalid file type	1- define invalid construct
		2- run
			Define invalid (audio or frame) construct	Frame f1
				{
				path: "E:\images\frame.png"
				}
				extract audio from f1 format mp3 as vaudio
					User should find no changes happened	Audio extraction failed invalid file type	Action at line 7: Can't extract audio from a non-video type	Passed

`Action at line 7: Can't extract audio from a non-video type`								

TC_extract_audio_03	Valid file type Missing format	1- define valid construct
		2- run
			Define valid (video) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				
				extract audio from v1 as vaudio
					User should find no changes happened	Audio extraction failed no format	line 6:22 mismatched input 'as' expecting 'format'	Passed

`line 6:22 mismatched input 'as' expecting 'format'`								

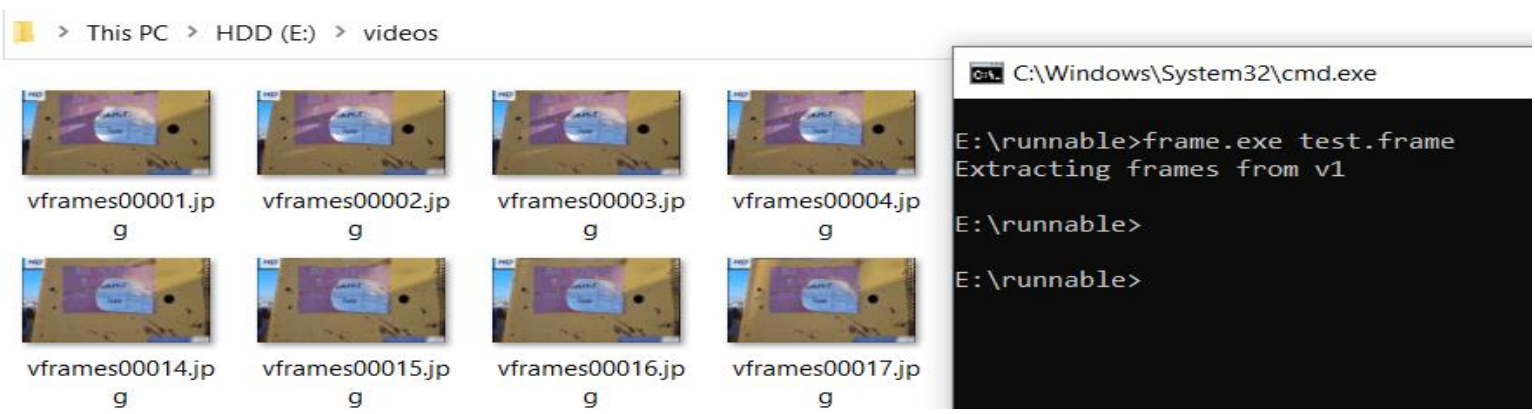
TC_extract_audio_04	Valid file type Invalid audio format	1- define valid construct
		2- pass invalid audio format
		3- run
			Define valid (video) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				extract audio from v1 format png as audio
					User should find no changes happened	Audio extraction failed no format	line 6:22 mismatched input 'as' expecting 'format'	Passed

`line 6:29 mismatched input 'png' expecting AUDIO_FORMAT`								

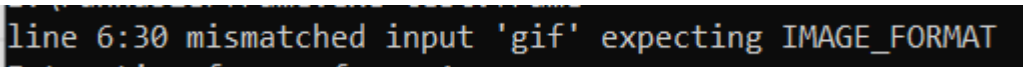
Test Scenario ID	**Test Scenario Description**
TS_Frame_02	Verify extract frames from video functionality

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

TC_extract_frames_01	Valid image format	1- pass valid image format
		2- run
			Define valid(video) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				
				extract frames from v1 format jpg as vframes
					User should be able to find that frames has been extracted in the given format	Frames extracted successfully	Extracting frames from v1	Passed



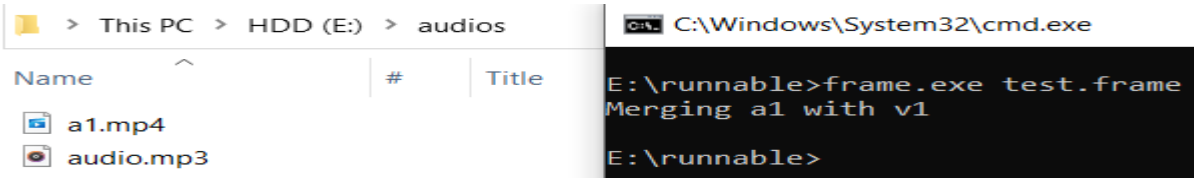
TC_extract_frames_02	Invalid image format	1- pass invalid image format 2- run	Define valid(video) construct	extract frames from file_construct format gif as extracted_frames	User should be able to find that frames has not extracted in the given invalid format	Frames extraction failed	line 6:30 mismatched input 'gif' expecting IMAGE_FORMAT	Passed



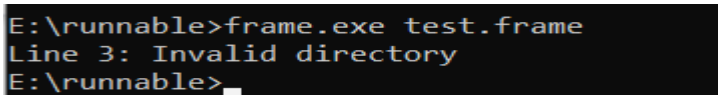
Test Scenario ID	**Test Scenario Description**
TS_Frame_03	Verify merge audio with video functionality

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

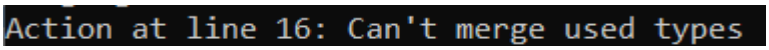
TC_merge_audio_with_video_01	Valid audio file
	Valid video file
	No global options
		1- define valid audio construct
		2- define valid video construct
		3- run
			Define valid (audio) construct
			Define valid (video) construct
				Audio a1
				{
				path: "E:\audios\audio.mp3"
				}
				Video v1
				{
				path: "E:\videos\video.mp4"
				}
				merge a1 with v1
					User should be able to find both audio and video merged into one file which is a1	Two files are merged successfully	Merging a1 with v1	Passed

TC_merge_audio_with_video_02	Valid audio file
	Valid video file
	Define invalid output directory in global options
		1- define invalid directory in global options
		2- define valid audio construct
		3- define valid video construct
		4- run
			Define valid (audio) construct
			Define valid (video) construct
				Options
				{
				output: ":E\merged_files"
				}
				Audio a1
				{
				path: "E:\audios\audio.mp3"
				}
				Video v1
				{
				path: "E:\videos\video.mp4"
				}
				merge a1 with v1
					User should be able to find no changes happened	Two files are merging failed invalid directory	Line 3: Invalid directory	Passed

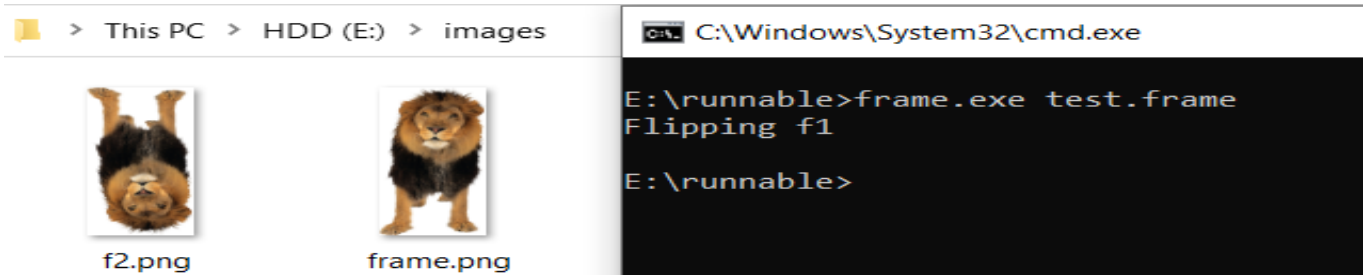
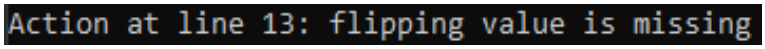
TC_merge_audio_with_video_03	invalid file(frame)
	Valid video file
		1- define invalid audio construct
		2- define valid video construct
		3- run
			Define invalid (audio) construct
			Define valid (video) construct
				Frame f1
				{
				path: "E:\images\frame.png"
				}
				merge f1 with v1 as merged
					User should be able to find no changes happened	Merging two files failed invalid file type	Action at line 16: Can't merge used types	Passed

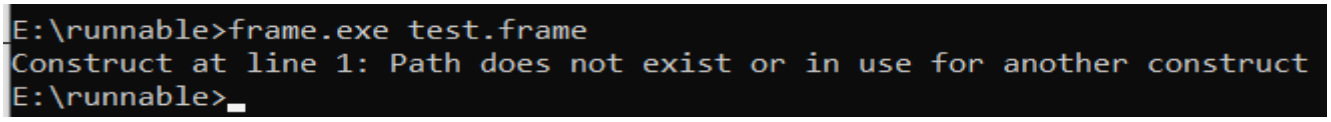
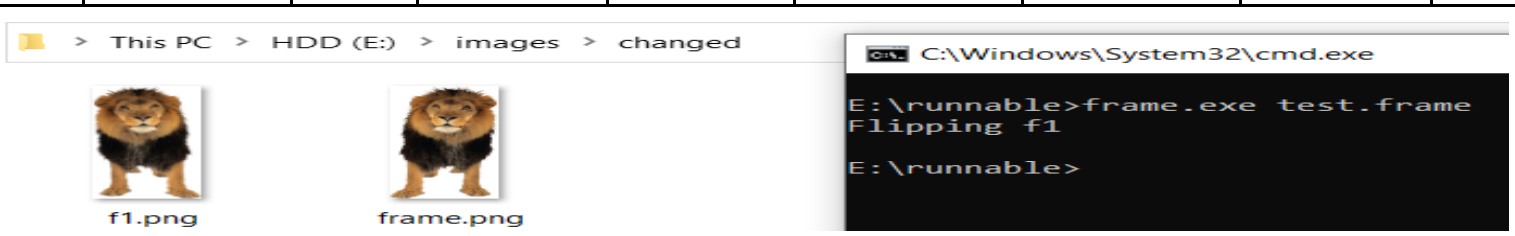
Test Scenario ID	Test Scenario Description
TS_Frame_04	Verify flip video or frame vertically or horizontally functionality

Test Case ID	Test Case Description	Test Steps	Preconditions	Test Data	Postconditions	Expected Result	Actual Result	Status

TC_flip_video_or_frame_01	Vertical flip	1- pass vertical flip option
		2- run
			Define valid (video or frame) construct	Frame f1
				{
				path: "E:\images\frame.png"
				}
				flip f1 vflip as f2
					User should be able to see f2 flipped vertically	File flipped vertically successfully	Flipping f1	Passed

								
TC_flip_video_or_frame_02	Missing flip option	1- pass empty flip option 2- run	Define valid (video or frame) construct	flip f1 as f2	User should be able to see no changes	Flipped failed no flip option	Action at line 13: flipping value is missing	Passed
								

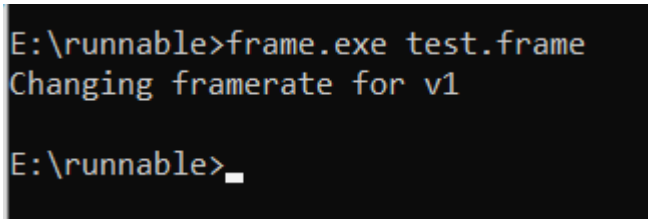
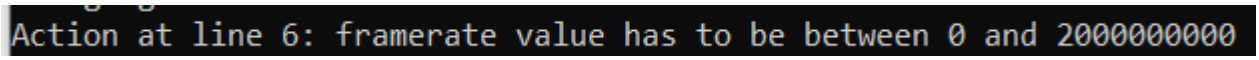
TC_flip_video_or_frame_03	Vertical flip	1- pass vertical flip option
		2- pass invalid file type
		2- run
			Define valid (video or frame) construct	Frame f1
				{
				path: "E:\images\frame.mp4"
				}
				flip f1 vflip as f2
					User should see that no changes happened	Framed flipping failed construct not exists	Construct at line 1: Path does not exist or in use for another construct	Passed

								
TC_flip_video_or_frame_04	Horizontal flip	1- pass horizontal flip option 2- pass valid path 2- run	Define valid (video or frame) construct	flip f1 hflip save to "E:\images\changed\"	User should be able to see that f1 has been flipped horizontally(mirror) and saved in the specified path	Frame flipped horizontally successfully		Passed
								

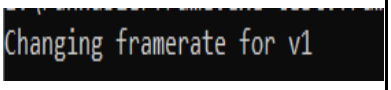
Test Scenario ID	Test Scenario Description
TS_Frame_05	Verify Changing the framerate of a video file to a specific value.

Test Case ID	Test Case Description	Test Steps	Preconditions	Test Data	Postconditions	Expected Result	Actual Result	Status

TC_set_framerate_video_01	Valid (video) file
	Valid framerate value
		1- pass valid framerate value
		2- run
			Define valid (video) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				
				set framerate for v1 15 as v2
					User should be able to find v2 saved with specified framerate	Video framerate changed successfully	Changing framerate for v1	Passed

								
TC_set_framerate_video_02	Valid (video) file Invalid framerate value	1- pass invalid framerate value 2- run	Define valid (video) construct	set framerate for file_construct 2000000001 as new_file	User should be able to find that no changes in video framerate	Changing video framerate failed invalid framerate	Action at line 6: framerate value has to be between 0 and 2000000000	Passed
								

TC_set_framerate_video_03	invalid (audio) file
	Valid framerate value
		1- pass valid framerate value
		2- run
			Define invalid (audio) construct	Audio v1
				{
				path: "E:\videos\video.mp4"
				}
				
				set framerate for v1 15 as v2
					User should be able to find that no changes in video framerate	Changing video framerate failed Invalid file type	Action at line 6: Can't change framerate for audio type	Passed

								
TC_set_framerate_video_04	Valid (video) file Valid framerate value	1- pass valid framerate value 2- pass valid path 3- run	Define valid (video) construct	set framerate for v1 15 as v2 save to "E:\videos\changed\"	User should be able to find v2 saved with specified framerate in specified path	Video framerate changed successfully and saved to new path	Changing framerate for v1	Passed
								

Test Scenario ID	**Test Scenario Description**
TS_Frame_06	Verify setting the volume of a video/audio file to a specific value.

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

TC_set_volume_video_or_audio_01	Valid (video or audio) file
	Valid volume value
		1- pass valid volume value
		2- run
			Define valid (video or audio) construct	Audio a1
				{
				path: "E:\audios\audio.mp3"
				}
				
				set volume for a1 300 as a2
					User should be able to find a2 volume set as specified	Volume set successfully	Changing volume for a1	Passed

Changing volume for a1								
TC_set_volume_video_or_audio_02	Valid (video or audio) file Missing volume value	1- pass empty volume value 2- run	Define valid (video or audio) construct	set volume for a1 as a2	User should find no change in file volume	Volume set failed missing parameter	line 6:18 missing INT at 'as'	Passed
Action at line 33: volume value is missing								
TC_set_volume_video_or_audio_03	Valid (video or audio) file Invalid volume value	1- pass invalid volume value 2- run	Define valid (video or audio) construct	set volume for a1 10000000 as a2	User should find no change in file volume	Volume set failed invalid volume	Action at line 6: volume value has to be between -9999999.0 and 9999999.0	Passed
Action at line 33: volume value has to be between 0.0 and 9999999.0								

TC_set_volume_video_or_audio_04	Invalid (frame) file
	Valid volume value
		1- pass valid volume value
		2- run
			Define invalid (frame) construct	Frame f1
				{
				path: "E:\images\frame.png"
				}
				
				set volume for f1 300 as f2
					User should find no changes in volume	Volume set failed Invalid file type	Action at line 6: Can't change volume for frame type	Passed

Action at line 6: Can't change volume for frame type								

Test Scenario ID	**Test Scenario Description**
TS_Frame_07	Verify Scaling/Resizing a video/frame file according to the specified width and height values.

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

TC_scale_video_or_frame_01	Valid (video or frame) file
	Valid width
	Valid height
		1- pass valid width
		2- pass valid height
		3- run
			Define valid (video or frame) construct	Frame f1
				{
				path: "E:\images\frame.png"
				}
				
				scale f1 width 500 height 500 as f2
					User should be able to find f2 scaled as specified	f1 scaled successfully	Scaling f1	Passed

TC_scale_video_or_frame_02	Valid (video or frame) file Invalid width Invalid height	1- pass invalid width 2- pass invalid height 3- run	Define valid (video or frame) construct	scale f1 width 0 height 0 as f2	User should be able to find no scaling applied	Scaling failed invalid width and height	Action at line 6: width value has to be between 1 and 16000 Action at line 6: height value has to be between 1 and 16000	Passed

TC_scale_video_or_frame_03	Invalid (audio) file
	Valid width
	Valid height
		1- pass valid width
		2- pass valid height
		3- run
			Define invalid (audio) construct	Audio a1
				{
				path: "E:\audios\audio.mp3"
				}
				
				scale a1 width 500 height 500 as a2
					User should be able to find no scaling applied	Scaling failed Invalid file type	Action at line 6: Can't scale audio type	Passed

```
E:\runnable>frame.exe test.frame
Scaling f1
```

```
Action at line 29: width or height value is missing
```

```
Action at line 6: Can't scale audio type
```

Test Scenario ID	**Test Scenario Description**
TS_Frame_08	Verify sobel filtering of a video according to the specified intensity.

Test Case ID	**Test Case Description**	**Test Steps**	**Preconditions**	**Test Data**	**Postconditions**	**Expected Result**	**Actual Result**	**Status**

TC_sobel_video_01	Valid (video or frame) file
	
	Valid intensity
		1- pass valid intensity 2-run	Define valid (video or frame) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				
				sobel v1 intensity 50
					User should be able to find v1 sobeled with the specified intensity	v1 sobeled successfully	f1 was filtered successfully	Passed

TC_sobel_video_02	Valid (video or frame) file
	
	Invalid intensity
		1- pass invalid intensity 2-run	Define valid (video or frame) construct	Video v1
				{
				path: "E:\videos\video.mp4"
				}
				
				sobel v1 intensity 65536
					User shouldn't be able to find any results as an error has occurred	Error message displays that the user has entered intensity value out of range	Action at line 6: sobel's intensity value has to be between 0.0 and 65535.0	Passed

```
E:\runnable>frame.exe test.frame
Sobelling v1...

E:\runnable>
```

```
Action at line 6: sobel's intensity value has to be between 0.0 and 65535.0
```