The Trace Project

Traits Mnemonics
grostaco@gmail.com

November 3, 2020

Abstract

Nowadays, reverse image search technology is becoming more and more ubiquitous. Images often are shared across online platforms, with little means to trace them back to their source. General-purpose reverse image lookup search engines are widely available, widely-known implementations are from web-crawlers like the google reverse image search, making it near impossible to sometimes get accurate results. This paper goes into detail about the Trace project.

1 Trace

Trace is an amalgamation of functionalities, mainly focusing on media frame extraction and processing.

Trace's primary goal is for efficient target-specific reverse image queries with extensibility for further contributions in other types of queries. For this purpose, Trace's design is to be modular; the essential functionality is called Core, and any specific-query addons will be called extensions.

The next section will discuss Trace's 'Core' functions, and how it interacts with its extensions.

2 Core

Trace was designed for a modular approach, where Trace's core has been built in such a way to support more OOP oriented paradigms, and still usable for lower-level languages without said OOP support directly from their compiler.

Trace's core current functionality is frame extraction, glibc's FILE* approach by using virtual jump tables along with permitting symbol overriding to grant some semblance of a higher-level language was adopted. The C functions exposed to the users are listed below.

2.1 Types

Trace provides an interface akin to that of FILE* called frameobject*, meaning that users are not encouraged to modify any fields within this aggregate type.

The type frameobject is an aggregate(struct) type that contains data required for libav packet send & receive, swscaling, error buffers, etc. Its fields are also highly subjected to change, backward compatibility will only be retained through exposed API calls unless stated otherwise. Information on each field is available in the current repository of Trace.[1]

Another prevalent type used in Trace's current code is struct __frame_frameobject_plus, which is a structure type containing __frame_frameobject (an alias of frameobject) and __internal_frame_jump_t, a virtual jump table for existing Trace's methods. The current design is a work in progress, as it is slowly fully adopting glibc's FILE* approach. This, too, is subjected to change in the future.

2.2 C Functions

Trace's C functions provide an interface to the opaque frameobject* type, ranging from opening a media file, extracting it, or skipping some frames. These are listed as follows.

frameobject* **frame_open**(const char* file)
Allocates a frameobject type as a handle to **file**. Returns a non-NULL address on success.

size_t **frame_extract**(frameobject* __frameobject, enum AVPixelFormat __pixfmt, enum AVCodecID __dstfmtid, ...)
Extracts frames from the file opened with frame_open in **__frameobject**, it will treat the extracted pixel format as specified by **__pixfmt** and convert it to the format specified by **__dstfmtid**.
frame_extract provides multiple ways to save the files, passed through its variadic argument

Option	Description
FRAME_ENDARG	Indicates when the argument list ends, this must always be passed to frame_extract
FRAME_NSAVE	The amount of maximum frames to save, another integer argument must be passed for the amount of frames requested
FRAME_BULKSAVE	Save all frames in one file
FRAME_SEPSAVE	Save all frames in different files

Table 1: frame_extract save options

int **frame_error**(frameobject* __frameobject)
Returns the latest error code from any frameobject methods invoked

size_t **frame_skip**(frameobject* __frameobject, size_t nframes)
Reads and discard up to **nframes**, synonymous with using the hidden flag FRAME_SKIP passed to **frame_extract**.

int **frame_close**(frameobject* __frameobject)
Frees all allocated resources from **frame_open**

2.3 Python C API wrapper functions

Lorem ipsum

2.4 Python Functions

Lorem ipsum

2.5 Contributions

Lorem ipsum

3 Extension

3.1 Trace C-Python API

Trace has a Python extension, allowing it to be called from any Python interpreter. The module is called Trace, further information on installing can be found on Trace's repository.

3.2 Trace's Discord Interface

Trace is planned to have an interface through discord, currently the Trace's discord bot is a standalone with no relations to the Core Trace currently.

4 Planned Usages

4.1 Developers

Trace is geared towards being a general-purpose reverse image lookup tool, making it possible for any extensions needing it for any specific-query to expand upon it. Trace is current a work in progress, but it is steadily gaining functionalities to serve as a subpar tool.

4.2 Users

Trace's initial purpose was to make a discord account which has oauth rights to respond to users and reverse-search specific queries (i.e movies, Japanese cartoons, etc.). But as the possibilities grew, Trace has been envisioned as a general-purpose tool suited for specific tasks.

5 Features

5.1 Currently Available Features

- Object oriented approach to media extraction in C.
- Python module for current Trace's Core functions.
- Python function for histogram conversion to a numpy-loadable file.
- Python function for comparing color histogram files.
- Discord account for oauth2 automation powered by discord.py, equipped with

basic functionalities.

5.2 Features currently being implemented

- Reworks on Python Modules to also being more object oriented to fit Python's paradigm
- Image to histogram conversion en masse
- Integrate discord bot's end to Trace's Core functions

5.3 Features planned

- "Bulk image" load to histograms
- Locality sensitive hashing
- Optimizations on bottlenecks wherever possible

6 Milestones

6.1 Milestone passed

6.1.1 Milestone A

- Decode arbitrary video formats to MJPEG or any picture format that can be represented as an image
- Convert MJPEG format files to histograms
- A Discord oauth2 account to automate generic responses

6.1.2 Milestone B

- Python C API module to provide facilities for video decode
- Complete the Discord bot's responses with more sophisticated messages (e.g., embeds)
- Merge and consolidate code into one repository
- A generic function for histogram conversion for both store and load

6.2 Milestone pending

6.2.1 Milestone C

- Host the Discord bot on a cloud server with marginal storage
- Check for efficiency and scalability of the current method
- Tie every functions together from each modular parts
- Add a how-to for installation of Trace on the official repository
- Doxygen documentation for every function