TLC 2017 EXCHANGE INNOVATION 交流 · 创新

FFmpeg滤镜开发 - 人脸识别

刘歧 OnVideo 联合创始人

- 现任职于OnVideo
- 业余参与维护与开发 FFmpeg
- 音视频流媒体爱好者

个人介绍

内容大纲

技术选择

契机

- 项目需要
- 视频图像识别技术火热
- 好奇视频图像识别实现
- 社区中很多人对相关技术有兴趣

技术选择 - 开源版

- dlib (Boost Software License)
- opencv (BSD License)
- openface (Apache 2.0 License)

技术选择 - 商业版

- qcloud
- SenseTime
- FaceUnity
- Face++
- 涂图
- 视诀
- ……

faceunity™

涂图 TUTUCLOUD

Face++ 旷视 人工智能开放平台

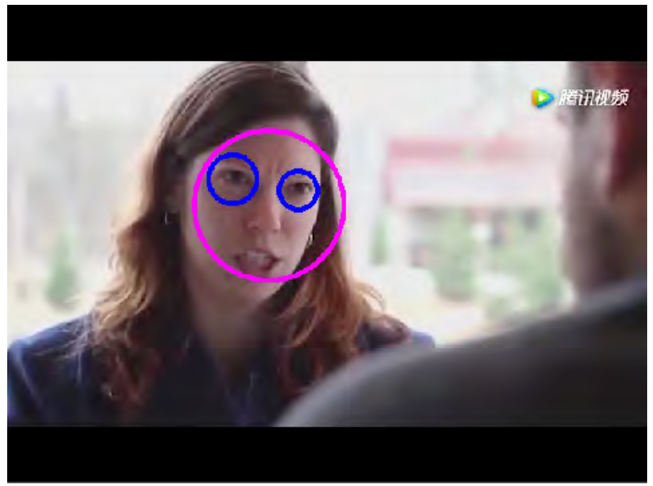
SENSETIME 商汤科技

技术选择 - 选择困难

- 众多选择，各有优劣势
- License 规避处理
- 基于 FFmpeg 原有的 OCV 滤镜处理
- 基于 FFmpeg 的 Frei0r (GPL v2)
- 自己封装一个so

Switch接口 - 可适配

- init 初始化各种参数
- object_detect 识别的对象
- get_rect 获得识别到的对象的画面范围
- finit 结束使用




```
liuqideMBP:1 liuqi$ ./DisplayImage a.jpg

This program demonstrates using the cv::CascadeClassifier class to detect objects (Face + eyes) in a video stream.
You can use Haar or LBP features.

x=[188], y =[127], width = [149], height = [149]
```

集成操作

集成操作 - AVFilter

- 参考 frei0r 滤镜

```
static int load_path(AVFilterContext *ctx, void **handle_ptr, const char *prefix, const char *name)
{
    char *path = av_asprintf("%s%s%s", prefix, name, SLIBSUF);
    if (!path)
        return AVERROR(ENOMEM);
    av_log(ctx, AV_LOG_DEBUG, "Looking for frei0r effect in '%s'.\n", path);
    *handle_ptr = dlopen(path, RTLD_NOW|RTLD_LOCAL);
    av_free(path);
    return 0;
}
```

```
static void *load_sym(AVFilterContext *ctx, const char *sym_name)
{
    Frei0rContext *s = ctx->priv;
    void *sym = dlsym(s->dl_handle, sym_name);
    if (!sym)
        av_log(ctx, AV_LOG_ERROR, "Could not find symbol '%s' in loaded module.\n", sym_name);
    return sym;
}
```

```
typedef f0r_instance_t (*f0r_construct_f)(unsigned int width, unsigned int height);
typedef void (*f0r_destruct_f)(f0r_instance_t instance);
typedef void (*f0r_deinit_f)(void);
typedef int (*f0r_init_f)(void);
typedef void (*f0r_get_plugin_info_f)(f0r_plugin_info_t *info);
typedef void (*f0r_get_param_info_f)(f0r_param_info_t *info, int param_index);
typedef void (*f0r_update_f)(f0r_instance_t instance, double time, const uint32_t *inframe, uint32_t *outframe);
typedef void (*f0r_update2_f)(f0r_instance_t instance, double time, const uint32_t *inframe1, const uint32_t *inframe2, const uint32_t *inframe3, uint32_t *outframe);
typedef void (*f0r_set_param_value_f)(f0r_instance_t instance, f0r_param_t param, int param_index);
typedef void (*f0r_get_param_value_f)(f0r_instance_t instance, f0r_param_t param, int param_index);
```

集成操作 - AVFilter

- AVFilter操作接口 filter_frame
- 接口操作对象 AVFrame *in
- 接口输出对象 AVFrame *out

```
static int filter_frame(AVFilterLink *inlink, AVFrame *in)
{
    Frei0rContext *s = inlink->dst->priv;
    AVFilterLink *outlink = inlink->dst->outputs[0];
    AVFrame *out;

    out = ff_get_video_buffer(outlink, outlink->w, outlink->h);
    if (!out) {
        av_frame_free(&in);
        return AVERROR(ENOMEM);
    }
    av_frame_copy_props(out, in);

    s->update(s->instance, in->pts * av_q2d(inlink->time_base) * 1000,
                    (const uint32_t *)in->data[0],
                    (uint32_t *)out->data[0]);

    av_frame_free(&in);

    return ff_filter_frame(outlink, out);
}
```

集成操作 - 多AVFilter联动

- 从一个AVFilter将参数传递至另一个AVFilter
- AVFrame中需要增加
  - object_x
  - object_y
  - object_width
  - object_height

```
static int filter_frame(AVFilterLink *inlink, AVFrame *in)
{
    Frei0rContext *s = inlink->dst->priv;
    AVFilterLink *outlink = inlink->dst->outputs[0];
    AVFrame *out;

    out = ff_get_video_buffer(outlink, outlink->w, outlink->h);
    if (!out) {
        av_frame_free(&in);
        return AVERROR(ENOMEM);
    }
    av_frame_copy_props(out, in);

    s->update(s->instance, in->pts * av_q2d(inlink->time_base) * 1000,
                  (const uint32_t *)in->data[0],
                  (uint32_t *)out->data[0]);

    av_frame_free(&in);

    return ff_filter_frame(outlink, out);
}
```

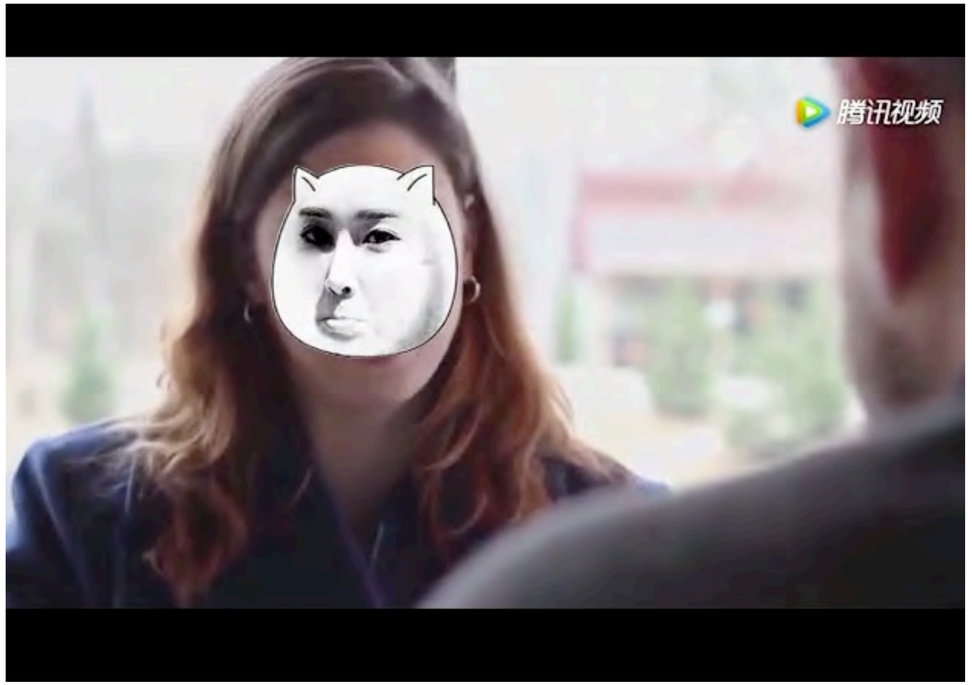
集成操作 - 多AVFilter联动

- 信息传递可通过 AVFrame 的 Metadata 进行

```
#define SET_META(key, value) \
    av_dict_set_int(&in->metadata, key, value, 0)
static int filter_frame(AVFilterLink *inlink, AVFrame *in)
{
    Frei0rContext *s = inlink->dst->priv;
    AVFilterLink *outlink = inlink->dst->outputs[0];
    AVFrame *out;
    int x = 0;
    int y = 0;
    int w = 0;
    int h = 0;;
    AVRational sar;

    out = ff_get_video_buffer(outlink, outlink->w, outlink->h);
    if (!out) {
        av_frame_free(&in);
        return AVERROR(ENOMEM);
    }
    av_frame_copy_props(out, in);

    s->update(s->instance, in->pts * av_q2d(inlink->time_base) * 1000, (const uint32_t *)in->data[0], (uint32_t *)out->data[0]);

    s->getrect(s->instance, &x, &y, &w, &h);
    if (!sar.num)
        sar.num = sar.den = 1;

    av_frame_make_writable(out);

    av_dict_set_int(&out->metadata, "lavfi.facedetect.x1", x, 0);
    av_dict_set_int(&out->metadata, "lavfi.facedetect.y1", y, 0);
    av_dict_set_int(&out->metadata, "lavfi.facedetect.h1", h, 0);
    av_dict_set_int(&out->metadata, "lavfi.facedetect.w1", w, 0);
    av_frame_free(&in);

    return ff_filter_frame(outlink, out);
}
```

集成操作 - 多AVFilter联动

- face detect filter 与 overlay filter联动效果
- ./ffmpeg -i input.mp4 -i picture.png -filter_complex "[0:v]facedetect[detect];[detect][1:v]overlay=x=detect_x:y=detect=y:w=detect_width=width:detect_height=height[output]" output.mp4



后需考虑

后续考虑 - 集成更多第三方

- Tensorflow 集成
- Pix2Pix 集成
- SRGAN 集成
- 优化 OpenCV + OpenCL

操作总结

操作总结

- 当出现选择困难症时就全选
- 然后抽象共性接口给 FFmpeg

END

感谢您的聆听.

-刘歧