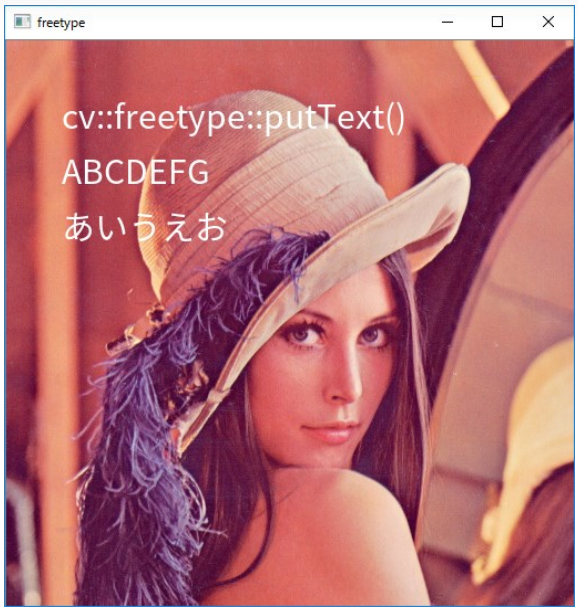


OpenCV显示中文

亚瑟的微风
github.com/zchrissirhcz

0x0 问题描述

使用OpenCV展示算法结果时，会遇到两类中文乱码/不显示问题：

1. cv::imshow的窗口标题，中文可能乱码
2. cv::putText中文显示为问号

0x1 让imshow窗口支持中文

cv::imshow中文标题乱码，通常只在Windows平台出现：和VS运行程序时在cmd输出中文乱码，本质一样的：Visual Studio的源码文件和编译时编码需要分别设定，不能混为一谈。

一个常见的、不是很好的实践方式，是把源码编码改为gb2312/gbk，则运行时cmd/imshow窗口的中文可以正常显示。

考虑到UTF-8的广泛使用，应当把源码文件按UTF-8编码保存；这导致的运行时cmd/imshow窗口乱码问题，可以通过传入编译参数，让cl.exe按GBK格式编译源码文件来避免。具体做法：

在VS工程属性->配置属性->C/C++->命令行->其它选项，填入

```
%(AdditionalOptions) /source-charset:utf-8 /execution-charset:gbk
```

或者，基于CMake构建的CMakeLists.txt里，设定：

```
if(CMAKE_C_COMPILER_ID STREQUAL "MSVC")
  set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} /source-charset:utf-8 /execution-charset:gbk")
  set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} /source-charset:utf-8 /execution-charset:gb
endif()
```

0x2 让putText支持中文 - 基于Windows GDI

cv::putText无法正常显示中文，在Windows/Linux/MacOSX下都可能出现。对于Windows平台，在不重新编译OpenCV的情况下，有快速修复方案。

基于前一节的设定(源码文件按UTF-8保存，编译选项指定GBK编码编译)，使用cv::putText添加中文时，在Windows系统上并不能正确显示中文，显示的是一串问号？？：

解决办法是用Windows API里的HDC系列函数来完成字符绘制，封装后提供的接口为 `cv::putTextZH()`，保存为cv_puttextzh.h文件，内容如下：

```
// cv::putTextZH(), the enhanced version for cv::putText, support drawing

#ifndef CV_PUTTEXTZH_H
#define CV_PUTTEXTZH_H

#include "opencv2/opencv.hpp"

//-----------------------------------------------------------------------
// declaration
//-----------------------------------------------------------------------

namespace cv {
    static void putTextZH(cv::Mat& dst, const char* str, cv::Point org, cv::Scala
}

//-----------------------------------------------------------------------
// implementation
//-----------------------------------------------------------------------

#define NOMINMAX
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#undef NOMINMAX
#undef WIN32_LEAN_AND_MEAN

static void GetStringSize(HDC hDC, const char* str, int* w, int* h)
{
    SIZE size;
    GetTextExtentPoint32A(hDC, str, strlen(str), &size);
    if (w != 0) *w = size.cx;
    if (h != 0) *h = size.cy;
}

namespace cv {

    void putTextZH(cv::Mat& dst, const char* str, cv::Point org, cv::Scalar color
    {
        CV_Assert(dst.data != 0 && (dst.channels() == 1 || dst.channels() == 3));
        int x, y, r, b;
        if (org.x > dst.cols || org.y > dst.rows) return;
        x = org.x < 0 ? -org.x : 0;
        y = org.y < 0 ? -org.y : 0;
        LOGFONTA lf;
        lf.lfHeight = -fontSize;
        lf.lfWidth = 0;
        lf.lfEscapement = 0;
        lf.lfOrientation = 0;
        lf.lfWeight = 5;
        lf.lfItalic = italic;   //斜体
        lf.lfUnderline = underline; //下划线
        lf.lfStrikeOut = 0;
        lf.lfCharSet = DEFAULT_CHARSET;
        lf.lfOutPrecision = 0;
        lf.lfClipPrecision = 0;
        lf.lfQuality = PROOF_QUALITY;
        lf.lfPitchAndFamily = 0;
        strcpy_s(lf.lfFaceName, fn);
        HFONT hf = CreateFontIndirectA(&lf);
        HDC hDC = CreateCompatibleDC(0);
        HFONT hOldFont = (HFONT)SelectObject(hDC, hf);
        int strBaseW = 0, strBaseH = 0;
        int singleRow = 0;
        char buf[1 << 12];
        strcpy_s(buf, str);
        char* bufT[1 << 12];  // 这个用于分隔字符串后剩余的字符，可能会超出。
        //处理多行
```

```
        {
            int nnh = 0;
            int cw, ch;
            const char* ln = strtok_s(buf, "\n", bufT);
            while (ln != 0)
            {
                GetStringSize(hDC, ln, &cw, &ch);
                strBaseW = std::max(strBaseW, cw);
                strBaseH = std::max(strBaseH, ch);
                ln = strtok_s(0, "\n", bufT);
                nnh++;
            }
            singleRow = strBaseH;
            strBaseH *= nnh;
        }
        if (org.x + strBaseW < 0 || org.y + strBaseH < 0)
        {
            SelectObject(hDC, hOldFont);
            DeleteObject(hf);
            DeleteObject(hDC);
            return;
        }
        r = org.x + strBaseW > dst.cols ? dst.cols - org.x - 1 : strBaseW - 1;
        b = org.y + strBaseH > dst.rows ? dst.rows - org.y - 1 : strBaseH - 1;
        org.x = org.x < 0 ? 0 : org.x;
        org.y = org.y < 0 ? 0 : org.y;
        BITMAPINFO bmp = { 0 };
        BITMAPINFOHEADER& bih = bmp.bmiHeader;
        int strDrawLineStep = strBaseW * 3 % 4 == 0 ? strBaseW * 3 : (strBaseW * 3 +
        bih.biSize = sizeof(BITMAPINFOHEADER);
        bih.biWidth = strBaseW;
        bih.biHeight = strBaseH;
        bih.biPlanes = 1;
        bih.biBitCount = 24;
        bih.biCompression = BI_RGB;
        bih.biSizeImage = strBaseH * strDrawLineStep;
        bih.biClrUsed = 0;
        bih.biClrImportant = 0;
        void* pDibData = 0;
        HBITMAP hBmp = CreateDIBSection(hDC, &bmp, DIB_RGB_COLORS, &pDibData, 0, 0);
        CV_Assert(pDibData != 0);
        HBITMAP hOldBmp = (HBITMAP)SelectObject(hDC, hBmp);
        //color.val[2], color.val[1], color.val[0]
        SetTextColor(hDC, RGB(255, 255, 255));
        SetBkColor(hDC, 0);
        //SetStretchBltMode(hDC, COLORONCOLOR);
        strcpy_s(buf, str);
        const char* ln = strtok_s(buf, "\n", bufT);
        int outTextY = 0;
        while (ln != 0)
        {
            TextOutA(hDC, 0, outTextY, ln, strlen(ln));
            outTextY += singleRow;
            ln = strtok_s(0, "\n", bufT);
        }
        uchar* dstData = (uchar*)dst.data;
        int dstStep = dst.step / sizeof(dstData[0]);
        unsigned char* pImg = (unsigned char*)dst.data + org.x * dst.channels() +
        unsigned char* pStr = (unsigned char*)pDibData + x * 3;
        for (int tty = y; tty <= b; ++tty)
        {
            unsigned char* subImg = pImg + (tty - y) * dstStep;
            unsigned char* subStr = pStr + (strBaseH - tty - 1) * strDrawLineStep;
            for (int ttx = x; ttx <= r; ++ttx)
            {
                for (int n = 0; n < dst.channels(); ++n) {
                    double vtxt = subStr[n] / 255.0;
                    int cvv = vtxt * color.val[n] + (1 - vtxt) * subImg[n];
                    subImg[n] = cvv > 255 ? 255 : (cvv < 0 ? 0 : cvv);
                }
                subStr += 3;
                subImg += dst.channels();
            }
        }
        SelectObject(hDC, hOldBmp);
        SelectObject(hDC, hOldFont);
        DeleteObject(hf);
        DeleteObject(hBmp);
        DeleteDC(hDC);
    }
}

#endif // CV_PUTTEXTZH_H
```

具体使用例子：

```
#include "cv_puttextzh.h"

int main() {
    cv::Mat image = cv::imread("colorhouse.png");
    cv::putTextZH(
       image,
       "Hello, OpenCV很赞",
       cv::Point(10, image.rows / 2),
       CV_RGB(255, 255, 255),
       20
    );
    cv::imshow("原图", image);
    cv::waitKey(0);

    return 0;
}
```

0x3 让putText支持中文 - 基于freetype

尽管Linux/MacOSX下默认编码是UTF-8，imshow窗口的中文可以正常显示；然而cv::putText对于中文字符仍然不能正常显示（一连串问号）。幸运的是，opencv_contrib里的freetype模块解决了这个问题，提供了替代cv::putText的文字绘制API。

ubuntu20.04下编译带freetype的opencv

需要先安装freetype和harfhuzz，在Ubuntu 20.04下的命令为：

```
sudo apt install libfreetype-dev libharfbuzz-dev
```

可通过 `sudo apt-cache madison libfreetype-dev` 辅助验证版本信息。若未安装freetype和harfbuzz则会在后续编译opencv的cmake阶段遇到报错：

```
364 -- Checking for module 'freetype2'
365 --   No package 'freetype2' found
366 -- Checking for module 'harfbuzz'
367 --   No package 'harfbuzz' found
368 -- freetype2:   NO
```

拉取opencv_contrib仓库，里面包含freetype模块，然后切换contrib库版本，并在重新编译opencv时指定contrib仓库路径：

```
git clone https://gitee.com/mirrors/opencv_contrib
cd opencv_contrib
git checkout -b 4.5.1 4.5.1
```

编译脚本compile.sh:

```
#!/bin/bash
set -x
set -e

rm -rf build
mkdir -p build
cd build

LOG="../cmake.log"
touch $LOG
rm $LOG

exec &> >(tee -a "$LOG")

cmake .. \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/home/zz/lib/opencv/4.5.2-pre \
  -DOPENCV_EXTRA_MODULES_PATH=/home/zz/work/opencv_contrib-4.5/modules \
  -DBUILD_LIST=core,imgcodecs,imgproc,freetype \
  -DCV_DISABLE_OPTIMIZATION=ON \
  -WITH_CUDA=OFF \
  -WITH_VTK=OFF

make -j8
make install

cd ..
```

编译安装带freetype模块的opencv后，在自己的工程里使用，样例代码：

```
#include <opencv2/opencv.hpp>
#include <opencv2/freetype.hpp> //Note1:头文件

int main() {
    cv::Mat image = cv::imread("colorhouse.png");

    //Note2:创建freetype指针，和加载ttf字体
    //思源字体，downloaded from https://www.google.com/get/noto/help/cjk/
    cv::Ptr<cv::freetype::FreeType2> ft2 = cv::freetype::createFreeType2();
    cv::String ttf_pathname = "NotoSansCJKjp-Medium.otf";
    ft2->loadFontData(ttf_pathname, 0);

    //Note3:执行绘制
    ft2->putText(
        image,
        "Hello, OpenCV 很赞",
        cv::Point(20, 200),
        30,
        CV_RGB(0, 0, 255),
        cv::FILLED,
        cv::LINE_AA,
        true
    );

    cv::imwrite("colorhouse_with_text.png", image);

    return 0;
}
```

效果：

win10下编译带freetype的opencv

实测发现Windows下编译带freetype的opencv，比在Linux下要繁琐，一些cmake的设定需要修改后才能正确编译，记录如下：

安装pkg-config.exe

下载 pkg-config-lite，sourceforge.net/project ...

解压后添加bin目录到系统PATH环境变量

编译安装freetype2

下载 freetype-2.10.4 源码：download.savannah.gnu.org ...

解压后创建 build/vs2019-x64.cmd，内容：

```
@echo off

set BUILD_DIR=vs2019-x64
if not exist %BUILD_DIR% md %BUILD_DIR%
cd %BUILD_DIR%
cmake ../.. -G "Visual Studio 16 2019" -A x64 ^
    -DCMAKE_INSTALL_PREFIX=e:/lib/freetype/2.10.4 ^
    -DFT_WITH_BZIP2=OFF ^
    -DFT_WITH_HARFBUZZ=OFF ^
    -DFT_WITH_PNG=OFF ^
    -DFT_WITH_ZLIB=OFF
cmake --build . --config Debug
cmake --build . --config Release
cmake --install . --config Debug
cmake --install . --config Release
cd ..
pause
```

双击执行

编译安装harfbuzz

下载 harfbuzz 2.7.4 源码 github.com/harfbuzz/har ...

解压后创建 build/vs2019-x64.cmd 脚本，内容：

```
@echo off

set BUILD_DIR=vs2019-x64
if not exist %BUILD_DIR% md %BUILD_DIR%
cd %BUILD_DIR%
cmake ../.. -G "Visual Studio 16 2019" -A x64 ^
    -DCMAKE_INSTALL_PREFIX=e:/lib/harfbuzz/2.7.4 ^
    -DCMAKE_PREFIX_PATH=E:/lib/freetype/2.10.4 ^
    -DCMAKE_DEBUG_POSTFIX=d ^
    -DHB_BUILD_SUBSET=ON ^
    -DHB_BUILD_UTILS=OFF ^
    -DHB_HAVE_DIRECTWRITE=OFF ^
    -DHB_HAVE_FREETYPE=ON ^
    -DHB_HAVE_GDI=OFF ^
    -DHB_HAVE_GLIB=OFF ^
    -DHB_HAVE_GOBJECT=OFF ^
    -DHB_HAVE_GRAPHITE2=OFF ^
    -DHB_HAVE_ICU=OFF ^
    -DHB_HAVE_INTROSPECTION=OFF ^
    -DHB_HAVE_UNISCRIBE=OFF

cmake --build . --config Debug
cmake --build . --config Release

cmake --install . --config Debug
cmake --install . --config Release

cd ..
pause
```

双击执行

编译安装opencv

确保本地已有 opencv_contrib 仓库，并切换到 4.5.1 tag; 修改 opencv_contrib/freetype/CMakeLists.txt，把原来的：

```
ocv_check_modules(FREETYPE freetype2)
ocv_check_modules(HARFBUZZ harfbuzz)
```

修改为：

```
if (WIN32)
  find_package(Freetype REQUIRED)
  #注:尝试过在调用cmake时指定HARFBUZZ_ROOT，发现无效。暂时在本文件里硬编码路径，实测有效
  set(HARFBUZZ_ROOT "E:/lib/harfbuzz/2.7.4" CACHE PATH "harfbuzz install root dir")
  find_path(HARFBUZZ_INCLUDE_DIRS
    NAMES hb-ft.h PATH_SUFFIXES harfbuzz
    HINTS ${HARFBUZZ_ROOT}/include)
  find_library(HARFBUZZ_LIBRARIES
      NAMES harfbuzz
      HINTS ${HARFBUZZ_ROOT}/lib)
  find_package_handle_standard_args(HARFBUZZ
      DEFAULT_MSG HARFBUZZ_LIBRARIES HARFBUZZ_INCLUDE_DIRS)

else()
  ocv_check_modules(FREETYPE freetype2)
  ocv_check_modules(HARFBUZZ harfbuzz)
endif()
```

（否则无法正确找到freetype和harfbuzz导致不会生成freetype目标）

确保本地已有 opencv 仓库，并切换到 master 分支（2021-03-14 编译时会显示为4.5.2-pre版

本）；

创建 build/vs2019-x64.cmd 脚本，内容（因为windows编译太慢，里面还关了很多其他选项）：

```
@echo off

set BUILD_DIR=vs2019-x64
if not exist %BUILD_DIR% md %BUILD_DIR%
cd %BUILD_DIR%


cmake -G "Visual Studio 16 2019" -A x64 ../.. ^
  -DCMAKE_BUILD_TYPE=Release ^
  -DCMAKE_INSTALL_PREFIX=E:/lib/opencv/4.5.2-pre ^
  -DBUILD_LIST=core,imgcodecs,imgproc,highgui,photo,gapi,freetype ^
  -DOPENCV_EXTRA_MODULES_PATH=G:/dev/opencv_contrib-4.5.1/modules ^
  -DFREETYPE_INCLUDE_DIRS=E:/lib/freetype/2.10.4/include/freetype2 ^
  -DFREETYPE_LIBRARY_DEBUG=E:/lib/freetype/2.10.4/lib/freetyped.lib ^
  -DFREETYPE_LIBRARY_RELEASE=E:/lib/freetype/2.10.4/lib/freetype.lib ^
  -DHARFBUZZ_INCLUDE_DIRS=E:/lib/harfbuzz/2.7.4/include/harfbuzz ^
  -DHARFBUZZ_LIBRARY_DEBUG=E:/lib/harfbuzz/2.7.4/lib/harfbuzzd.lib ^
  -DHARFBUZZ_LIBRARY_Release=E:/lib/harfbuzz/2.7.4/lib/harfbuzz.lib ^
  -DBUILD_TIFF=OFF ^
  -DBUILD_OPENJPEG=OFF ^
  -DBUILD_JASPER=OFF ^
  -DBUILD_JPEG=ON ^
  -DBUILD_OPENEXR=OFF ^
  -DBUILD_WEBP=OFF ^
  -DBUILD_TBB=OFF ^
  -DBUILD_IPP_IW=OFF ^
  -DBUILD_ITT=OFF ^
  -DWITH_AVFOUNDATION=OFF ^
  -DWITH_CAP_IOS=OFF ^
  -DWITH_CAROTENE=OFF ^
  -DWITH_CPUFEATURES=OFF ^
  -DWITH_EIGEN=OFF ^
  -DWITH_FFMPEG=OFF ^
  -DWITH_GSTREAMER=OFF ^
  -DWITH_GTK=OFF ^
  -DWITH_IPP=OFF ^
  -DWITH_HALIDE=OFF ^
  -DWITH_VULKAN=OFF ^
  -DWITH_INF_ENGINE=OFF ^
  -DWITH_NGRAPH=OFF ^
  -DWITH_JASPER=OFF ^
  -DWITH_OPENJPEG=OFF ^
  -DWITH_JPEG=ON ^
  -DWITH_WEBP=OFF ^
  -DWITH_OPENEXR=OFF ^
  -DWITH_TIFF=OFF ^
  -DWITH_OPENVX=OFF ^
  -DWITH_GDCM=OFF ^
  -DWITH_TBB=OFF ^
  -DWITH_HPX=OFF ^
  -DWITH_PTHREADS_PF=OFF ^
  -DWITH_V4L=OFF ^
  -DWITH_CLP=OFF ^
  -DWITH_OPENCL=OFF ^
  -DWITH_OPENCL_SVM=OFF ^
  -DWITH_ITT=OFF ^
  -DWITH_PROTOBUF=OFF ^
  -DWITH_IMGCODEC_HDR=OFF ^
  -DWITH_IMGCODEC_SUNRASTER=OFF ^
  -DWITH_IMGCODEC_PXM=OFF ^
  -DWITH_IMGCODEC_PFM=OFF ^
  -DWITH_QUIRC=OFF ^
  -DWITH_ANDROID_MEDIANDK=OFF ^
  -DWITH_TENGINE=OFF ^
  -DWITH_ONNX=OFF ^
  -DBUILD_opencv_apps=OFF ^
  -DBUILD_ANDROID_PROJECTS=OFF ^
  -DBUILD_ANDROID_EXAMPLES=OFF ^
  -DBUILD_DOCS=OFF ^
  -DBUILD_EXAMPLES=OFF ^
  -DBUILD_PACKAGE=OFF ^
  -DBUILD_PERF_TESTS=OFF ^
  -DBUILD_TESTS=OFF ^
  -DBUILD_FAT_JAVA_LIB=OFF ^
  -DBUILD_ANDROID_SERVICE=OFF ^
  -DBUILD_JAVA=OFF ^
  -DBUILD_OBJC=OFF ^
  -DENABLE_PRECOMPILED_HEADERS=OFF ^
  -DENABLE_FAST_MATH=OFF ^
  -DCV_TRACE=OFF ^
  -DBUILD_opencv_java=OFF ^
  -DBUILD_opencv_objc=OFF ^
  -DBUILD_opencv_js=OFF ^
  -DBUILD_opencv_ts=OFF ^
  -DBUILD_opencv_python2=OFF ^
  -DBUILD_opencv_python3=OFF ^
  -DBUILD_opencv_dnn=OFF ^
  -DBUILD_opencv_videoio=OFF ^
  -DBUILD_opencv_calib3d=OFF ^
  -DBUILD_opencv_flann=OFF ^
  -DBUILD_opencv_objdetect=OFF ^
  -DBUILD_opencv_stitching=OFF ^
  -DBUILD_opencv_ml=OFF ^
  -DCV_DISABLE_OPTIMIZATION=ON ^
  -DWITH_VTK=OFF ^
  -DWITH_CUDA=OFF ^
  -DCMAKE_FIND_DEBUG_MODE=FALSE

cmake --build . --config Debug -j4
cmake --build . --config Release -j4

cmake --install . --config Debug
cmake --install . --config Release

cd ..

pause
```

双击执行

在个人项目中集成

和在Linux下使用带freetype的opencv的cv::putText方法一样，样例代码：

```
#include <opencv2/opencv.hpp>
#include <opencv2/freetype.hpp> //Note1:头文件
```

```
int main() {
    cv::Mat image = cv::imread("colorhouse.png");

    //Note2:创建freetype指针，和加载ttf字体
    cv::Ptr<cv::freetype::FreeType2> ft2 = cv::freetype::createFreeType2();
    ft2->loadFontData("c:/Windows/Fonts/simfang.ttf", 0); //加载字库文件

    // Put Text
    ft2->putText(image, "cv::freetype::putText()", cv::Point(50, 50), 30, cv::Scalar
    ft2->putText(image, "OpenCV很赞", cv::Point(50, 100), 30, cv::Scalar(255, 255, 25
    ft2->putText(image, "あいうえお", cv::Point(50, 150), 30, cv::Scalar(255, 255, 255),

    cv::imshow("测试", image);
    cv::waitKey(0);

    cv::imwrite("colorhouse_with_text.png", image);

    return 0;
}
```

实际测试发现，按照本文先前的VS工程属性设定，freetype显示的中文是方框：

解决方法：需要设定VS的执行编码为utf-8，才能确保freetype正确绘制中文，也就是：
```
%(AdditionalOptions) /source-charset:utf-8 /execution-charset:utf-8
```

效果：

MacOSX下的编译安装

和 Windows / Linux 步骤类似，参考 “5.月色的部落格 - 为OPENCV添加freetype支持并显示中文字符”

0x4 总结

对于opencv的中文编码显示问题，在Windows平台可以快速做到： 1. 修改工程属性，源码作为UTF-8对待，编译阶段则转为GBK编码 2. 使用基于GDI API封装的cv::putTextZH()

而对于 Linux/MacOSX，虽然理论上也有类似于GDI的平台API可以调用，一方面本人对这些API不了解，另一方面基于opencv contrib仓库里的freetype模块的编译也非常快速，因此推荐基于freetype模块重新编译opencv。

对于Windows平台来说，基于freetype重新编译OpenCV的过程则稍微曲折一些（freetype和harfbuzz的 `find_package()` 过程需要人工干预，很ugly），按照我贴出的步骤做下来putText可以正确显示中文，不过imshow的窗口中文标题会再次陷入乱码。

0x5 References

编辑于 2021-03-15 08:46

UTF-8 OpenCV Microsoft Visual Studio 2019

写下你的评论...

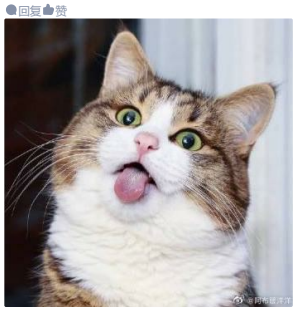
5 条评论
默认
最新

Cucibala
问一下为啥我编出来的不带 opencv2/freetype 这个库呢完全复制的上面的内容
2021-11-11
回复 赞

亚瑟的微风
作者
freetype 是 opencv_contrib 里的模块
2021-11-11
回复 1

渣渣图
NotoSansCJKjp-Medium.otf 这个文件如何获取呢?
2021-09-25
回复 赞

渣渣图
https://github.com/tweeeety/font-noto-sans
2021-09-25
回复 赞



橘子猪
学习下 

2021-03-15
回复 赞

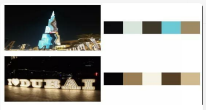
文章被以下专栏收录

移动端计算机视觉
C/C++，图像处理，SIMD优化

推荐阅读

如何安装opencv_contrib及解决其安装编译问题

1、背景最近在实现一个基于opencv3的自动人脸识别项目，主要是使用了cv2.face模块自带的三种人脸识别算法，分别是cv2.face.EigenFaceRecognizer_... cv2.face.FisherFaceRecognizer ...



基于OpenCV查找图像中最常见的颜色

从零学习 OpenCV 4

【从零学习OpenCV 4】opencv_contrib扩展模块...

OpenCV图像处理|1.2 加载、显示、修改、保存图像

人工智能技术干货

小白学视觉 发表于从零学习计...

小白学视觉 发表于搜罗最好玩...

Mengcius