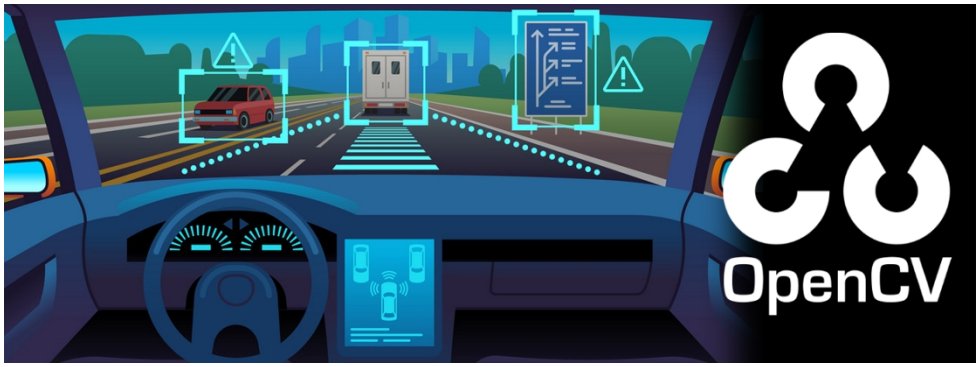
案例：表情包滤镜制作

本案例需要在桌面交互式环境下操作



案例概要

对于社交网络、短时频剪辑应用而言，表情包滤镜 (表情贴图) 一直是一个旺盛的需求。**表情包** 是一种利用图片来表示感情的一种方式。表情包是在社交软件活跃之后，形成的一种流行文化，表情包流行于互联网上面，基本人人都会发表情包。在移动互联网时期，人们以时下流行的明星、语录、动漫、影视截图为素材，配上一系列相匹配的文字，用以表达特定的情感。2017 年 7 月 18 日，教育部、国家语委在北京发布《中国语言生活状况报告(2017)》，表情包入选 2016 年度中国媒体十大新词。近年来，表情包除了应用于 IM 聊天工具的谈话交流以外，更逐步被应用到视频剪辑上面，将表情符号粘贴到视频中的人物头像上，譬如：近期经常应用在自制视频上的社会人表情包效果。

视频与摄像头媒体

在前面的实验中，我们一直通过 `cv2.imread` 函数读取静态图片。实际上，对于视频文件的处理方式与静态图片是一样的。首先，对于 Python 而言，视频与静态图片一样，都是一个对象。该对象可以是存储在本地的一段视频数据，也可以是来自特定摄像头的实时视频流。其次，对于 OpenCV 而言，视频实际上仅仅是多张连续的静态图片的集合。因此，对于视频的处理相当于对多张静态文件执行批量处理。

读取本地视频

我们可以直接通过 `cv2.VideoCapture` 函数调用视频对象，该函数可以直接填写视频名称直接加载本地视频文件作为输入参数，譬如：

```
cap = cv2.VideoCapture("./data/test.mp4")
```

读取本地摄像头

同时，我们也可以通过摄像头捕获实时图像数据，以摄像头代号作为输入参数， `0` 为默认摄像头，笔记本内建摄像头一般为 `0` ，譬如：

```
cap = cv2.VideoCapture(0)
```

读取网络摄像头

通常，摄像头使用 RTSP 或 HTTP 协议来传输视频。IP 摄像头网址流的格式一般是：
`rtsp://192.168.1.64/1` ，因此，可以通过以下代码实现使用 OpenCV 从网络摄像头获取视频流：

```
capture = cv2.VideoCapture('rtsp://192.168.1.64/1')
```

由于大多数 IP 摄像头都有用于访问视频的用户名和密码。在这种情况下，必须在网址流中提供凭据，如下所示：

```
capture = cv2.VideoCapture('rtsp://[用户名]:[密码]@192.168.1.64/1')
```

另外，各种型号的 IP 摄像头可能还会通过 URL 指定编码格式、通道等参数，具体以相关摄像头说明书或软件/手机应用程序说明为准，示例如下：

```
url = 'rtsp://admin:123456@192.168.1.216/H264?ch=1&subtype=0'
cap = cv2.VideoCapture(url)
```

案例目标

在本案例中，我们将使用本章中讨论的技术构建一个表情过滤器。

通过 OpenCV 的 `cv2.VideoCapture` 函数调用一段本地视频数据，并将我们自定的一个表情包图片粘贴到视频人物的头像上。最终效果如下所示：

案例详细操作步骤

1. 导入 OpenCV 和 NumPy 模块。
2. 定义表情包滤镜函数，通过使用一张表情图片，来替换视频中出现的人脸。其中函数包括三个输入参数：

   - **image**：表示视频每一帧
   - **cascadeFile**：级联分离器的路径
   - **emojiFile**：表情图片路径

   表情包函数分为如下几个步骤：

   A. 使用 `cv2.imread` 函数加载表情图片，确保函数中传递 `-1` 作为第二个参数，因为我们要读取图像中的 alpha 通道，负责图像的透明度。
   B. 使用 `cv2.cvtColor` 函数，将传入将帧进行灰度变换。
   C. 使用 `cv2.CascadeClassifier` 函数加载级联分类器。
   D. 使用 `haarCascade.detectMultiScale` 函数分类器检测帧中的人脸，引入 `if-else` 条件判断：

      - 如果没有检测到人脸则返回 `None`
      - 如果检测到人脸，则采用 `for` 循环，从 `haarCascade.detectMultiScale` 检测到的每个人脸轮廓中提取 `x`，`y`，`w`，`h` 参数。根据 `w`，`h` 参数，使用 `cv2.resize` 函数将表情图的尺寸转换为与检测到的人脸轮廓一致的宽度（`w`）和高度（`h`）。完成表情图尺寸调整后，我们将需要用表情图号替换面部。这就是 `alpha` 通道需要发挥作用的地方。在 `alpha` 通道中，`100%` 透明区域的值为 `0`（黑色区域），我们将图像中检测到人脸的区域的背景透明度设置为 `100%`，用表情图中透明度不为 `0` 的区域进行填充。你可以使用以下代码作为参考：

        ```
        (image[y:y+h, x:x+w])[np.where(emoji_resized[:,:,3]!=0)] = (emoji_resized
        [:,:,:3])[np.where(emoji_resized[:,:,3]!=0)]
        ```

   E. 表情包滤镜函数最后返回使用表情图填充后的视频帧。
3. 对表情包滤镜函数内的参数进行逐一指定，首先定义 `cascadeFile` 级联模型和 `emojiFile` 表情图路径。参考路径分别为：

   - `./data/haarcascade_frontalface_default.xml`
   - `./data/emoji.png` 注意：以数据实际存储路径为准。

4. 指定 `image`，这实际上是从视频文件中逐一抽取出来的每个独立帧。使用 `cv2. videoccapture` 函数创建视频对象 `cap`，以用于替换人脸的视频文件路径作为输入参数。参考路径为：`./data/test.mp4`；注意：以实际数据存储路径为准。
5. 创建一个 `while` 循环，在循环中，使用 `cap.read()` 函数，从 `cap` 对象中捕获帧。这将返回两个值
   - 一个返回值表示帧捕获是否成功
   - 另一个返回值表示捕获的帧
6. 在 `image` 中嵌入一个 `if-else` 条件判断：
   - 如果 `cap.read()` 的返回值为 `True`，意味着帧被成功读取，将它传递给 `emojiffilter` 函数，并使用 OpenCV 输出视频
   - 否则，中断循环

   你可以使用以下代码作为参考:

```
while True:

  # 捕获帧
  ret, frame = cap.read()

  emojiFilterResult = None

  # 查看帧是否捕获成功
  if ret == True:

      # 捕获成功，则调用过滤函数
      emojiFilterResult = emojiFilter(frame, haarCascade, emoji)
  else:
      break

  if emojiFilterResult is None:
      continue
  else:
      cv2.imshow("Emoji Filter", emojiFilterResult)
      k = cv2.waitKey(25)
      if k == 27:
          break
```

7. 使用 `cap.release()` 与 `cv2.destroyAllWindows()` 函数释放摄像头并关闭窗口。

练习题

> - **本案例所示的输出是使用预先录制的视频，而不是网络摄像头生成的。尝试在本地实验环境上，使用网络摄像头，通过使用表情图替换自己的头像，来生成各种有趣的表情包滤镜。**

案例小结

完成本案例后，你已经准备好了自己的表情过滤器。这可能不是一个很好看的过滤器，但这是一个开始，你可以通过随机切换不同的表情符号或根据是否出现微笑来让这一切变得更有趣。你还可以基于微笑检测在快乐和悲伤表情之间切换，来帮助你建立自己的情感检测表情过滤器。同样，你也可以添加其他透明图片到相框中，给你的

表情符号添加一些漂亮的妆容。

在本章中，我们开始了解为什么脸被认为是识别不同情绪、特征等的重要来源。我们还讨论了为什么我们不能有一个简单的人脸匹配算法，因为我们有大量的人脸需要进行识别。然后我们讨论了 Haar Cascades 的主题，并使用一个或多个级联分类器构建了几个应用程序：譬如微笑检测、正面人脸识别等等。我们还详细讨论了实验对与获取一组能精确检测特定图片中人脸的参数的重要性，这些参数将为我们提供使用级联分类器的最佳结果。然后，我们讨论了 GrabCut 技术，并建立了一个使用 ROI 区域掩膜的前后景分割应用。之后，我们对代码进行了修改，让用户可以对蒙版进行修正，以获得更好的效果。并且，在完成了前后景分离后进行背景融合替换。然后，我们介绍了如何使用 GrabCut 进行皮肤分割，最后以表情包过滤器来结束本章，它可以将表情包粘贴到从网络摄像头或视频文件的输入视频中的人脸上。

案例答案

1. 导入库

导入 OpenCV 模块和 numpy 模块。

In [1]:

```
import cv2
import numpy as np
```

2. 定义表情包滤镜函数

In [2]:

```
def emojiFilter(image, cascadeFile, emojiFile):

    # Step 1 加载表情图片，读取Alpha透明度通道
    emoji = cv2.imread(emojiFile,-1)

    # Step 2 将帧进行灰度变换
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Step 3 加载级联分类器
    haarCascade = cv2.CascadeClassifier(cascadeFile)

    # Step 4 使用分类器检测帧中的人脸
    detectedObjects = haarCascade.detectMultiScale(gray, 1.2, 3)

    # 如果没有检测到人脸则返回None
    if len(detectedObjects) == 0:
        return None

    # Step 5 如果检测到人脸，则使用表情图替换每一个人脸
    for bbox in detectedObjects:

        # 每个检测框的信息
        x, y, w, h = bbox

        # 根据矩形框转换表情图尺寸
        emoji_resized = cv2.resize(emoji, (w,h), interpolation = cv2.INTER_AREA)
        (image[y:y+h, x:x+w])[np.where(emoji_resized[:,:,3]!=0)] = \
        (emoji_resized[:,:,:3])[np.where(emoji_resized[:,:,3]!=0)]

    return image
```

3. 设置级联分类器和表情符号路径

注意，以数据实际存储路径为准。

In [3]:

```
# 设置输入输出路径
import os
base_path = os.environ.get("BASE_PATH",'../data/')
data_path = os.path.join(base_path + "lab5/")
result_path = "result/"
os.makedirs(result_path, exist_ok=True)

haarCascade = "./data/haarcascade_frontalface_default.xml"
emoji = "./data/emoji.png"
```

4. 加载视频

调用 cv2.VideoCapture 加载视频。注意，以数据实际存储路径为准。

In [4]:

```
cap = cv2.VideoCapture("./data/test.mp4")
```

5. 查看读取状态是都成功

In [5]:

```
if (cap.isOpened() == False):
    print("Error opening webcam")
```

6. 表情过滤

In [6]:

```
while True:

    # 捕获帧
    ret, frame = cap.read()

    emojiFilterResult = None

    # 查看帧是否捕获成功
    if ret == True:

        # 捕获成功，则调用过滤函数
        emojiFilterResult = emojiFilter(frame,
                                        haarCascade,
                                        emoji)
    else:
        break

    if emojiFilterResult is None:
        continue
    else:
        cv2.imshow("Emoji Filter", emojiFilterResult)
        k = cv2.waitKey(25)
        if k == 27:
            break
```

7. 释放摄像头关闭窗口

持续进行视频采集，可能最终导致流在终端上引起大量延迟，直到存储耗尽强制关闭该视频流，或视频流因其它原因自动中断。

In [7]:

```
cap.release()
cv2.destroyAllWindows()
```