6. Face effects

6.1. Introduction

MediaPipe is an open-source data stream processing machine learning application development framework developed by Google. It is a graph-based data processing pipeline used to build data sources in various forms, such as video, audio, sensor data, and any time series data. MediaPipe is cross-platform and can run on embedded platforms (such as Raspberry Pi), mobile devices (iOS and Android), workstations and servers, and supports mobile GPU acceleration. MediaPipe provides cross-platform, customizable ML solutions for real-time and streaming media. The core framework of MediaPipe is implemented in C++ and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include packets, streams, calculators, graphs, and subgraphs.

Features of MediaPipe:

- End-to-end acceleration: built-in fast ML inference and processing can be accelerated even on ordinary hardware.
- Build once, deploy anywhere: unified solution for Android, iOS, desktop/cloud, web, and IoT.
- Ready-to-use solution: cutting-edge ML solution that showcases the full capabilities of the framework.
- Free and open source: framework and solution under Apache 2.0, fully extensible and customizable.

6.2, Face effects

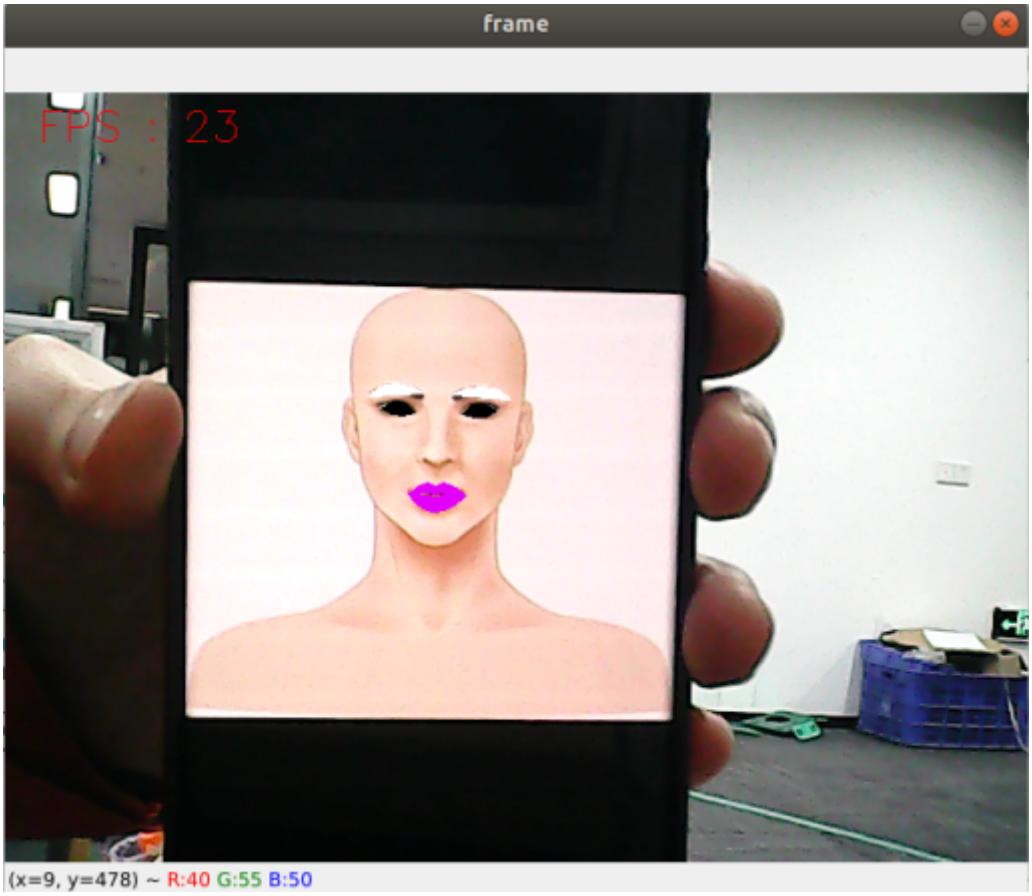
6.2.1, Start

Start the camera

```
roslaunch ascamera hp60c.launch
```

Terminal input,

```
roslaunch yahboomcar_mediapipe 06_FaceLandmarks.launch
```



6.2.2, Source code

Source code location:
/home/yahboom/ascam_ws/src/yahboomcar_mediapipe/scripts/06_FaceLandmarks.py

```
#!/usr/bin/env python3
# encoding: utf-8
import time
import dlib
import cv2 as cv
import numpy as np
from cv_bridge import CvBridge
from sensor_msgs.msg import Image
from std_msgs.msg import String
import rospy
import rospkg

class FaceLandmarks:
    def __init__(self, dat_file):
        self.hog_face_detector = dlib.get_frontal_face_detector()
        self.dlib_facelandmark = dlib.shape_predictor(dat_file)

    def get_face(self, frame, draw=True):
        gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
        self.faces = self.hog_face_detector(gray)
        for face in self.faces:
            self.face_landmarks = self.dlib_facelandmark(gray, face)
            if draw:
                for n in range(68):
                    x = self.face_landmarks.part(n).x
                    y = self.face_landmarks.part(n).y
```

```
                    cv.circle(frame, (x, y), 2, (0, 255, 255), 2)
                    cv.putText(frame, str(n), (x, y), cv.FONT_HERSHEY_SIMPLEX,
0.6, (0, 255, 255), 2)
        return frame

    def get_lmList(self, frame, p1, p2, draw=True):
        lmList = []
        if len(self.faces) != 0:
            for n in range(p1, p2):
                x = self.face_landmarks.part(n).x
                y = self.face_landmarks.part(n).y
                lmList.append([x, y])
                if draw:
                    next_point = n + 1
                    if n == p2 - 1: next_point = p1
                    x2 = self.face_landmarks.part(next_point).x
                    y2 = self.face_landmarks.part(next_point).y
                    cv.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)
        return lmList

    def get_lipList(self, frame, lipIndexlist, draw=True):
        lmList = []
        if len(self.faces) != 0:
            for n in range(len(lipIndexlist)):
                x = self.face_landmarks.part(lipIndexlist[n]).x
                y = self.face_landmarks.part(lipIndexlist[n]).y
                lmList.append([x, y])
                if draw:
                    next_point = n + 1
                    if n == len(lipIndexlist) - 1: next_point = 0
                    x2 = self.face_landmarks.part(lipIndexlist[next_point]).x
                    y2 = self.face_landmarks.part(lipIndexlist[next_point]).y
                    cv.line(frame, (x, y), (x2, y2), (0, 255, 0), 1)
        return lmList

    def prettify_face(self, frame, eye=True, lips=True, eyebrow=True,
draw=True):
        if eye:
            leftEye = self.get_lmList(frame, 36, 42)
            rightEye = self.get_lmList(frame, 42, 48)
            if draw:
                if len(leftEye) != 0: frame = cv.fillConvexPoly(frame,
np.mat(leftEye), (0, 0, 0))
                if len(rightEye) != 0: frame = cv.fillConvexPoly(frame,
np.mat(rightEye), (0, 0, 0))
        if lips:
            lipIndexlistA = [51, 52, 53, 54, 64, 63, 62]
            lipIndexlistB = [48, 49, 50, 51, 62, 61, 60]
            lipsUpA = self.get_lipList(frame, lipIndexlistA, draw=True)
            lipsUpB = self.get_lipList(frame, lipIndexlistB, draw=True)
            lipIndexlistA = [57, 58, 59, 48, 67, 66]
            lipIndexlistB = [54, 55, 56, 57, 66, 65, 64]
            lipsDownA = self.get_lipList(frame, lipIndexlistA, draw=True)
            lipsDownB = self.get_lipList(frame, lipIndexlistB, draw=True)
            if draw:
```

```
                if len(lipsUpA) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsUpA), (249, 0, 226))
                if len(lipsUpB) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsUpB), (249, 0, 226))
                if len(lipsDownA) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsDownA), (249, 0, 226))
                if len(lipsDownB) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lipsDownB), (249, 0, 226))
        if eyebrow:
            lefteyebrow = self.get_lmList(frame, 17, 22)
            righteyebrow = self.get_lmList(frame, 22, 27)
            if draw:
                if len(lefteyebrow) != 0: frame = cv.fillConvexPoly(frame,
np.mat(lefteyebrow), (255, 255, 255))
                if len(righteyebrow) != 0: frame = cv.fillConvexPoly(frame,
np.mat(righteyebrow), (255, 255, 255))
        return frame

class FaceEyeDetectionNode:
    def __init__(self):
        self.bridge = CvBridge()
        rospy.init_node("FaceEyeDetection", anonymous=False)
        self.face_landmarks =
FaceLandmarks(rospkg.RosPack().get_path("yahboomcar_mediapipe") +
"/scripts/file/shape_predictor_68_face_landmarks.dat")
        self.pub_rgb = rospy.Publisher("/FaceEyeDetection/image", Image,
queue_size=1)
        self.sub_image = rospy.Subscriber('/ascamera_hp60c/rgb0/image', Image,
self.image_callback, queue_size=1)
        self.pTime = 0

    def image_callback(self, msg):
        try:
            # Convert the ROS Image message to an OpenCV image
            frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding="bgr8")

            # Process the image
            frame = self.face_landmarks.get_face(frame, draw=False)
            frame = self.face_landmarks.prettify_face(frame, eye=True,
lips=True, eyebrow=True, draw=True)

            # Calculate and display FPS
            cTime = time.time()
            fps = 1 / (cTime - self.pTime)
            self.pTime = cTime
            text = f"FPS: {int(fps)}"
            cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0,
0, 255), 1)

            # Display the image
            cv.imshow('frame', frame)

            # Publish the processed image
            self.pub_rgb.publish(self.bridge.cv2_to_imgmsg(frame, "bgr8"))
```

```
            # Check for key press to quit
            if cv.waitKey(1) & 0xFF == ord('q'):
                rospy.signal_shutdown("User requested shutdown")
        except Exception as e:
            rospy.logerr("Could not process image: {e}")


if __name__ == '__main__':
    node = FaceEyeDetectionNode()
    rospy.spin()
    cv.destroyAllWindows()
```