Multimodal Large Model + Video Understanding (Text Version)

Before running the function, you need to close the App and large programs. For the closing method, refer to [4. Preparation] - [1. Manage APP control services].

1. Function Description

After the program runs, you can input questions about the video through the terminal. The large model will first record a video, then analyze the video content with the question, and finally reply with the answer to the question. You can also record a video of a specified duration first, then ask the large model questions related to the video content based on the video content.

2. Startup

Users with Jetson-Nano board version need to enter the docker container and input the following command. Orin board users can directly open the terminal and input the following command:

```
ros2 launch largemodel largemodel_control.launch.py text_chat_mode:=True
```

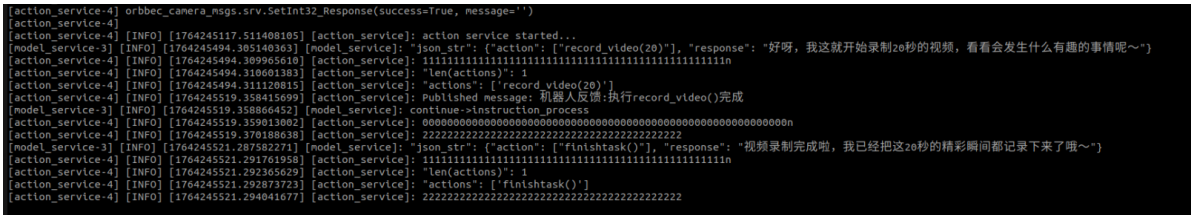
Then open a second terminal and input the following command:

```
ros2 run text_chat text_chat
```

Then input questions about visual understanding in the text_chat terminal. You can refer to the following examples:

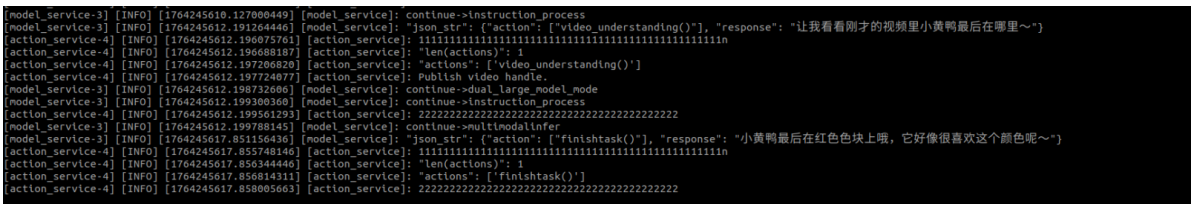
```
1. Record a 20-second video
```

After the buzzer beeps once,



After the buzzer beeps once, recording ends. Then based on this video content, input the following:

```
In the video just recorded, which color block is the little yellow duck on at the
end?
```



The location of the recorded and saved video files: For Orin board, videos are saved in /home/jetson. For Jetson-nano board, videos are saved in /root.

3. Core Code Analysis

3.1. Video Recording Function record_video

`Source code path: LargeModel_ws/src/largemodel/largemodel/action_service.py`

```
#Opens another terminal, starts the Record_Video node program, and passes the
parameter time_, which is the duration of the video recording
def record_video(self,time_):
    record_time_ = float(time_)
    cmd1 = f"ros2 run largemodel_arm Record_Video --ros-args -p time_:
{record_time_:.2f}"
    subprocess.Popen(
        [
            "gnome-terminal",
            "--title=recording",
            "--",
            "bash",
            "-c",
            f"{cmd1}; exec bash",
        ]
    )
#The Record_Video node program source code path is
LargeModel_ws/src/largemodel_arm/largemodel_arm/Record_Video.py. Mainly look at
the callback function for color images,
def get_RGBCallBack(self,msg):
    if self.beep_start==False:
        self.start_time = time.time()
        self.beep_start = True
        self.Arm.Arm_Buzzer_On()
        time.sleep(0.5)
        self.Arm.Arm_Buzzer_Off()
        self.record_done = False

    if self.record_done!=True:
        #Convert image topic data to image data
        rgb_image = self.rgb_bridge.imgmsg_to_cv2(msg,'bgr8')
        if self.out is None:
            #print("-----------")
            height, width, _ = rgb_image.shape
            fourcc = cv2.VideoWriter_fourcc(*'x264')  # Use MP4 encoder
            #Write image frames to video file
            self.out = cv2.VideoWriter(
                self.Path_video,
                fourcc,
                15,
                (width, height)
            )
        self.out.write(rgb_image)
        cv2.imshow('Video Recording', rgb_image)
        key = cv2.waitKey(1)
        elapsed = time.time() - self.start_time
        #Check recording duration, if greater than recording time, stop
recording
```

```
        if elapsed >= self.Time_ :
            cv2.destroyAllWindows()
            self.record_done = True
            self.out.release()
            self.Arm.Arm_Buzzer_On()
            time.sleep(0.5)
            self.Arm.Arm_Buzzer_Off()

 self.largemodel_arm_done_pub.publish(String(data='record_video_done'))
```

3.2. Video Understanding Function video_understanding

`Source code path: LargeModel_ws/src/largemodel/largemodel/action_service.py`

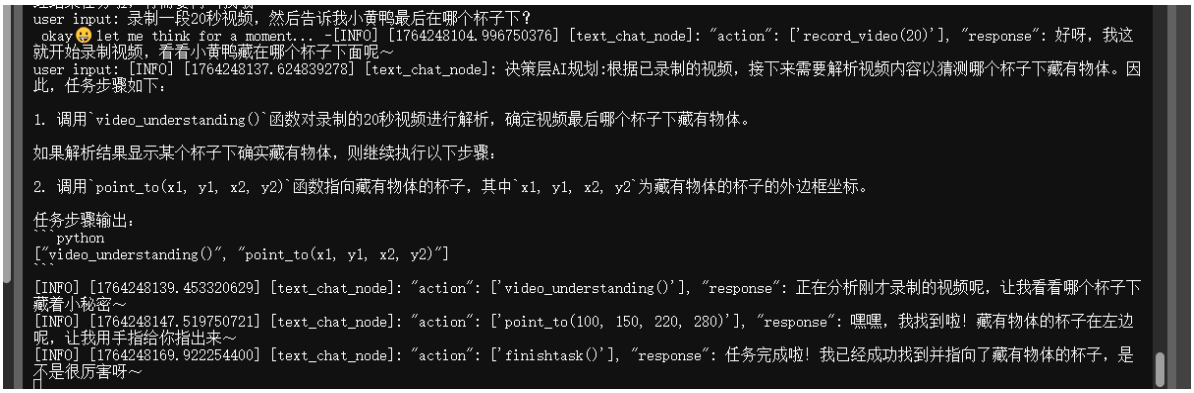
```
#After the model_service node receives video_handle, it will send the video to
the Alibaba Cloud large model platform, use the multimodal large model to analyze
the video content with the question, and finally return the result of the
question
def video_understanding(self):
    self.get_logger().info(
                "Publish video handle."
            )
    msg = String(data="video_handle")
    self.video_handle_pub.publish(
        msg
    )  # Normalize, publish video_handle topic, called by model_service to use
the large model
```

4. Combined with Robotic Arm

We can also combine robotic arm + video understanding to develop new gameplay. For example, prepare several opaque paper cups. After the video starts, we place an object under one of the opaque paper cups, then slowly change the positions of the paper cups. After the video recording ends, we ask the large model which cup the object is under at the end, and finally make the robotic arm point to that cup. To test this function, you can input in the terminal:

```
Let's play a game. First record a video, then guess which cup has the little
yellow duck at the end of the video?
```

Replace the little yellow duck with the actual object,



The task planning should be:

1. Call the record_video(20) function to record a 20-second video;
2. Call the video_understanding() function to analyze the recorded 20-second video to determine which cup hides the object at the end of the video.

If the analysis result shows that an object is indeed hidden under a certain cup, continue with the following steps:

3. Call the point_to(x1, y1, x2, y2) function to point to the cup hiding the object, where x1, y1, x2, y2 are the outer border coordinates of the cup hiding the object.