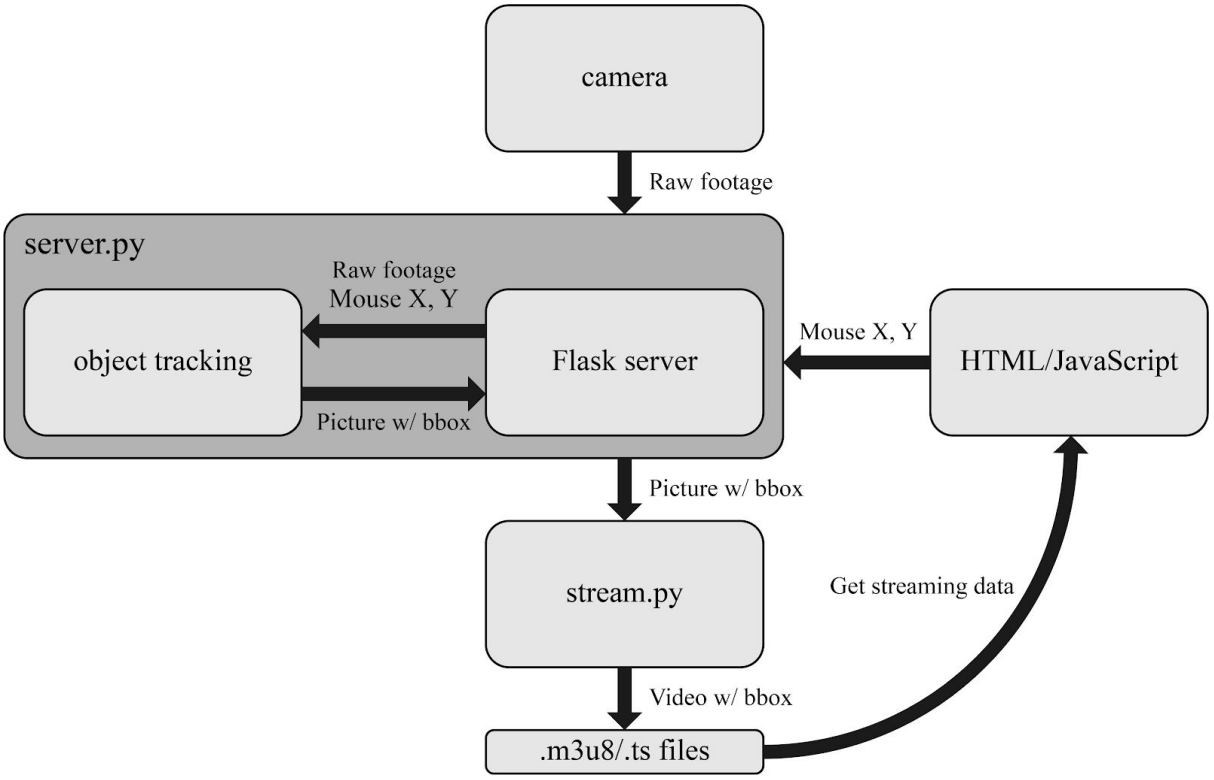
Video Streaming and Tracking Assignment 4

#1 Method



#1.1 Streaming

Step 1. Use a camera to capture frames and send them to the tracking model.
Step 2. Do object detection and tracking, then return the results to Flask server (*server.py*).
Step 3. HLS streaming server (*stream.py*) retrieves frame sequence from Flask server, then creates .ts files and .m3u8 file.
Step 4. Use <*video*> tag to show the video streaming in the browser.

#1.2 Mouse clicking event

Whenever a mouse click occurs, the browser sends the coordinate of mouse click to the Flask server via javascript.

1.3 Change bitrates (quality)

Step 1. After the HLS manifest has been loaded, we use JavaScript to get available quality levels from *Hls* object.
Step 2. Create a dropdown menu on the video player so that the user can change video quality.

GitHub: https://github.com/eugene87222/NCTU-Video-Streaming-and-Tracking

#2 Code explanation

#2.1 Mouse clicking event

(script/index.js)

```
function sendRequest(method, url) {
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open(method, url);
    xmlHttp.send(null);
    return xmlHttp.responseText;
}

const deselectButton = document.getElementById('deselect-button');
function deselect() {
    sendRequest('GET', 'http://localhost:8888/data?coor=deselect');
}
deselectButton.addEventListener('click', deselect);

function getClickCoordinate(event) {
    var x = event.clientX + window.pageXOffset -
videoContainer.offsetLeft;
    var y = event.clientY + window.pageYOffset -
videoContainer.offsetTop;
    var height = video.offsetHeight;
    var width = video.offsetWidth;
    sendRequest('GET',
'http://localhost:8888/data?coor='+x+','+y+','+height+','+width);
    event.preventDefault();
}
video.addEventListener('click', getClickCoordinate);
```

(server.py)

```
@app.route('/data')
def data():
    global tracks, trk_id
    coor = request.args.get('coor')
    if coor == 'deselect':
        trk_id = None
    else:
        x, y, h, w = coor.split(',')
        x, y = rescale(x, y, h, w)
        trk_id = select_track(x, y, target_cid, tracks)
    return json.dumps({'success': True}), 200,
{'ContentType':'application/json'}
```

Whenever a mouse clicking event occurs, the coordinate will be sent to *http://localhost:8888* where the Flask server is opened at.

#2.2 HLS streaming server

(stream.py)

```
def hls(*res):
    hls_dir = 'hls'
    clean_and_mkdir(hls_dir)
    codec_options = {
        'g': 10
    }
    hls = video.hls(
        Formats.h264(video='libx264', audio='aac', **codec_options),
        hls_time=1)
    hls.representations(*res)
    hls.output(os.path.join(hls_dir, 'hls.m3u8'))
```

The keyframe interval is set to 250 by default, which enforces *hls_time* to be 10 seconds. In order to present the object selection result at once, we have to make *hls_time* shorter. By setting the keyframe interval to 10, we can reduce *hls_time* to 1 second.

#2.3 Change bitrates (quality)

(index.html)

```
function setLevel(elem) {
  try {
    hls.currentLevel = parseInt(elem.getAttribute('level-idx'));
    var dropdownButton = document.getElementById('dropdown-button');
    dropdownButton.innerHTML = elem.innerHTML;
  }
  catch (e) {
    console.log(e);
  }
}

function createDropdownMenu() {
  var dropdown = document.getElementById('dropdown-content');
  hls.levels.forEach(function (item, index) {
    var node = document.createElement('div');
    node.innerHTML = `${item.height}p`;
    node.setAttribute('level-idx', index);
    node.setAttribute('onclick', 'setLevel(this)');
    dropdown.appendChild(node);
  });
  console.log('Dropdown menu created');
}

hls.on(Hls.Events.MANIFEST_LOADED, createDropdownMenu);
```

The dropdown menu is created after the HLS manifest has been loaded. When user chooses a quality level, we simply set the *currentLevel* attribute of the *HLS* object to the corresponding quality level index (-1 for automatic level selection).

#3 Problems & discussion

During the streaming method selection, we first tried out DASH. But since there are some codec issues on Firefox, we then switched to HLS.

While trying to support the object selection function, we had to split the .ts file into smaller time chunks in order to present the selection result at once. Thus we tried to modify the *hls_time*, but it still remains the same chunk size. We then found out that to split the chunk smaller, the video encoding has to include I-frames for the .ts file to inference. Thus we modified the default H.264 codec options to make it work.

#4 Reference

- UI design: https://freshman.tech/custom-html5-video/
- HLS streaming: https://video.aminyazdanpanah.com/python/start
- Flask server: https://towardsdatascience.com/video-streaming-in-web-browsers-with-opencv-flask-93a38846fe00

#5 Contribution

- 馬旭勃: HLS streaming server / report
- 顏劭庭: HLS streaming server / report
- 楊翔鈞: UI / integration / report