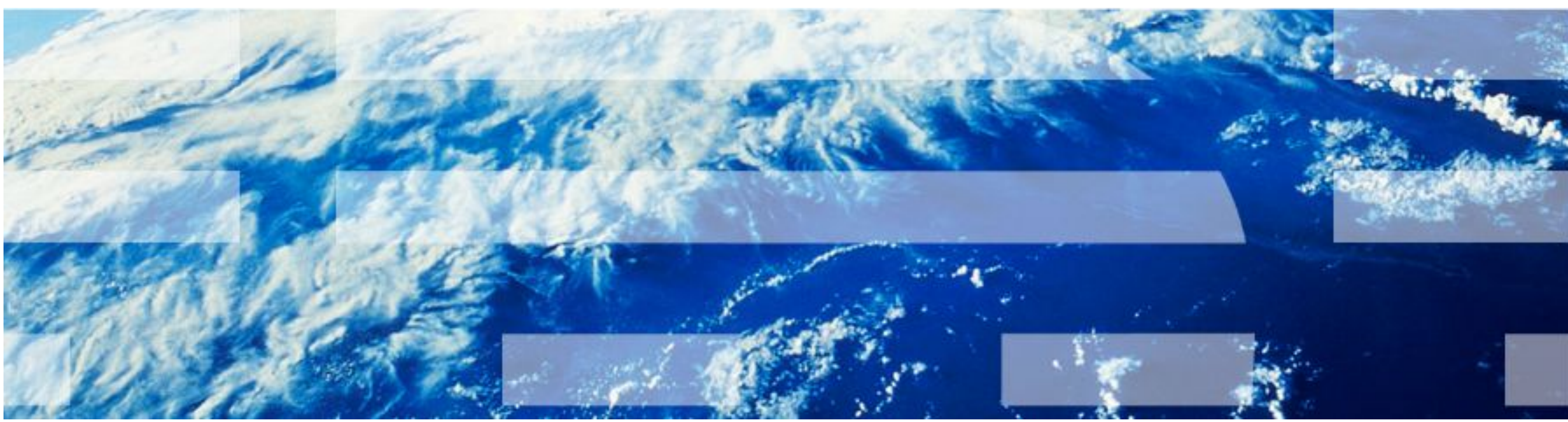
E6893 Big Data Analytics:

Video Object Segmentation based on Pixel-level Annotated dataset

Project ID:201912-13

Chong Hu/ch3467/EE
Yanlin Liu/yl4238/EE



Contents

Introduction

Video Object Segmentation

Goal: extracting foreground objects from video clips.

Application:

- video summarization/editing
- object tracking
- video action detection[3][4]
- autonomous driving
- etc…

Figure 1. separating foreground object(s) from the background region of a video[5]

Data Preprocess

Data

DAVIS 2016[2]

Densely **A**nnotated **VI**deo **S**egmentation

- 50 full HD video sequences

  > 3GB
- pixel-accurate ground-truth data provided for every video frame
- Contain occlusions, fast-motion, non-linear deformation and motion-blur

Figure 2. Sample images in DAVIS-2016 with annotation.[2]

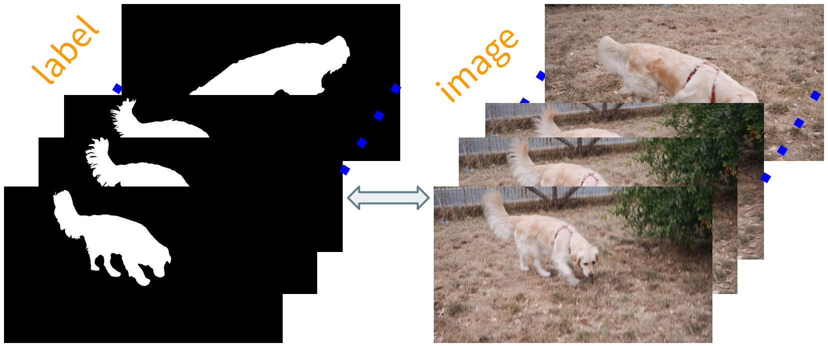
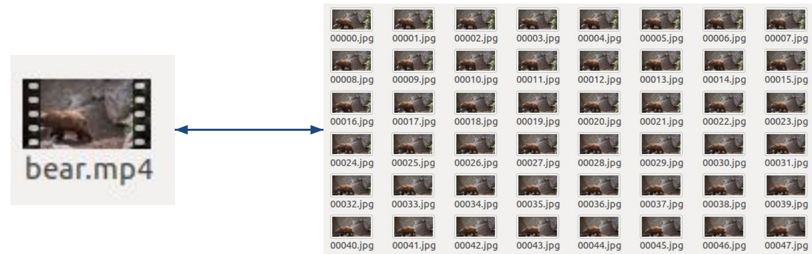


Figure 3. Image Sequence Data in DAVIS-2016.[2]

Data Preprocess

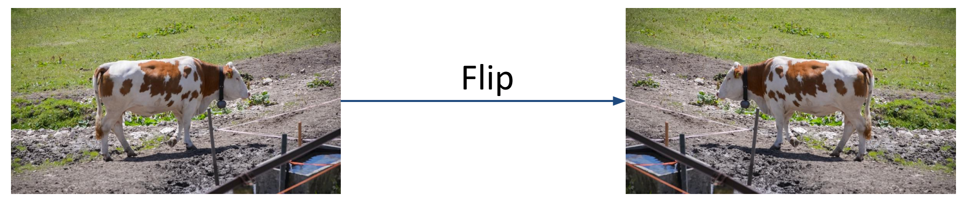
1. video <-> image

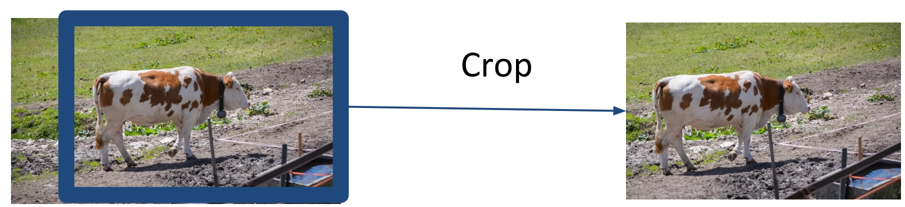
   480p 854x480x3



2. Data augmentation:

   Flip and Crop



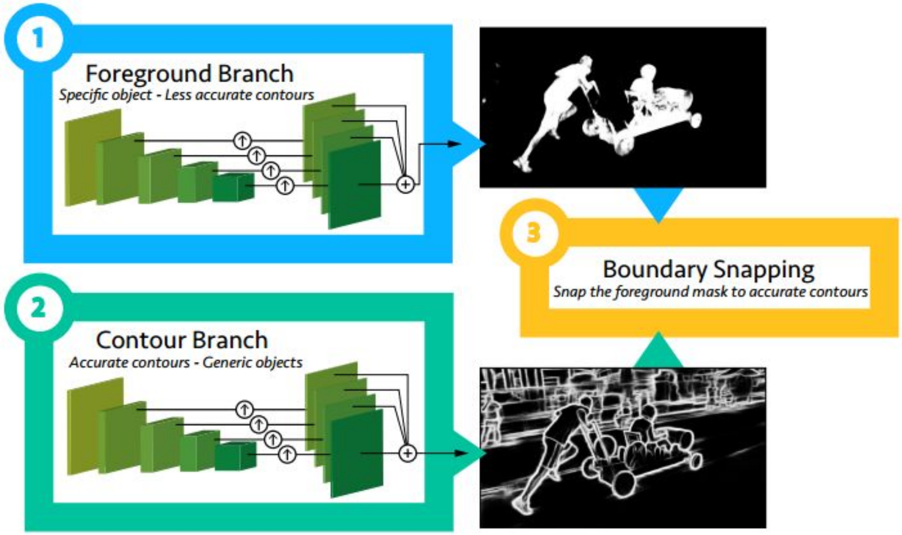


3. Train set vs. test set:
   a. Train set 30 video sequences
   b. Test set 20 video sequences

Model Construction

- Model:

  One-Shot VOS
  - Reasons:
    - Simpler: GPU limitation & easy to implement
    - Faster: One-Shot
    - Good result: ~80% in paper



- Model Structure
  - Parent network + Finetune (Transfer learning)
  - Two branch: Foreground & Contour
  - Loss function: imbalanced version of binary pixel-wise cross-entropy

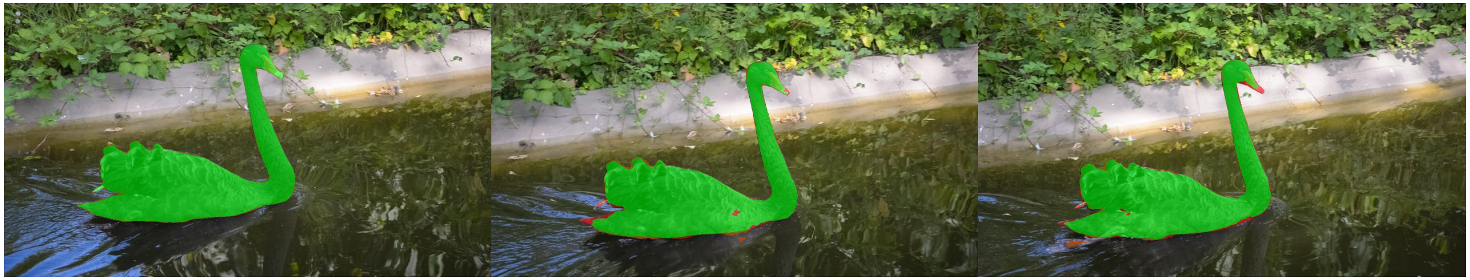
$$\mathcal{L}_{mod} = -\beta \sum_{j \in Y_+} \log P(y_j = 1|X) - (1-\beta) \sum_{j \in Y_-} \log P(y_j = 0|X) \quad (1)$$

where $\beta = |Y_-|/|Y|$.

Model Evaluation

- Visual evaluation:

  Good Result



frame 0 frame 19 frame 38

Bad Result

frame 0 frame 19 frame 38

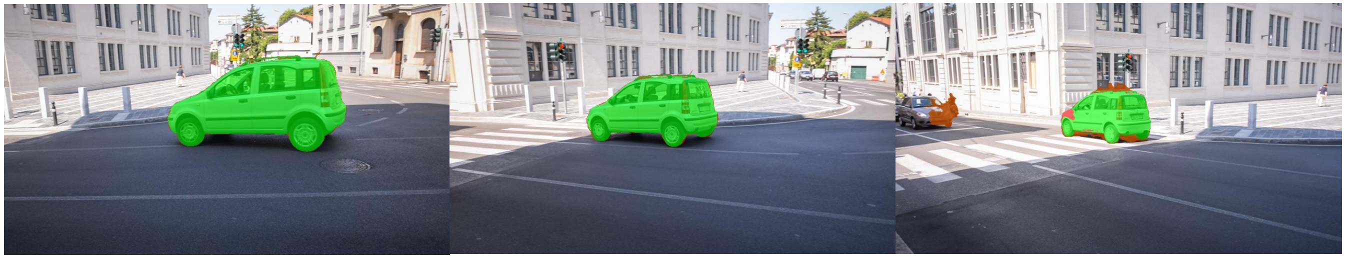
Possible reason: common object in pre-trained model <- already good

Not common and hard to learn <- improve not so large

Model Evaluation

- Visual evaluation:(more)

  Good Result



frame 0 frame 19 frame 38

Bad Result

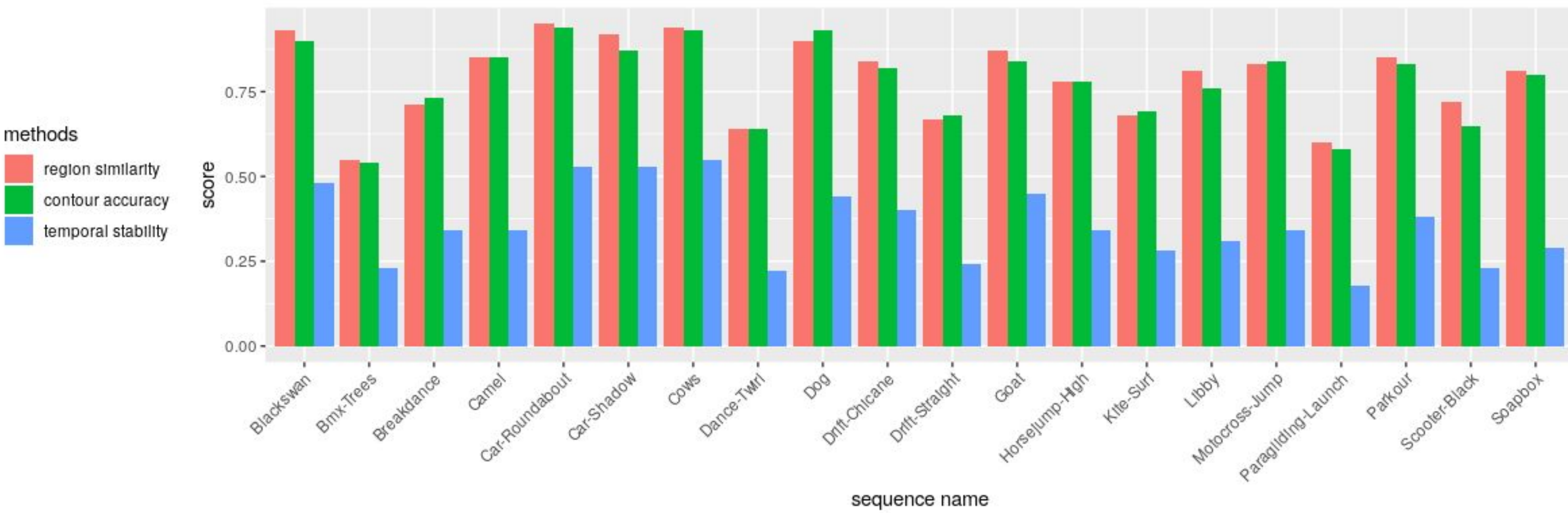
frame 0 frame 19 frame 38

Possible reason: common object in pre-trained model <- already good

Fast, high motion <- improve not so large

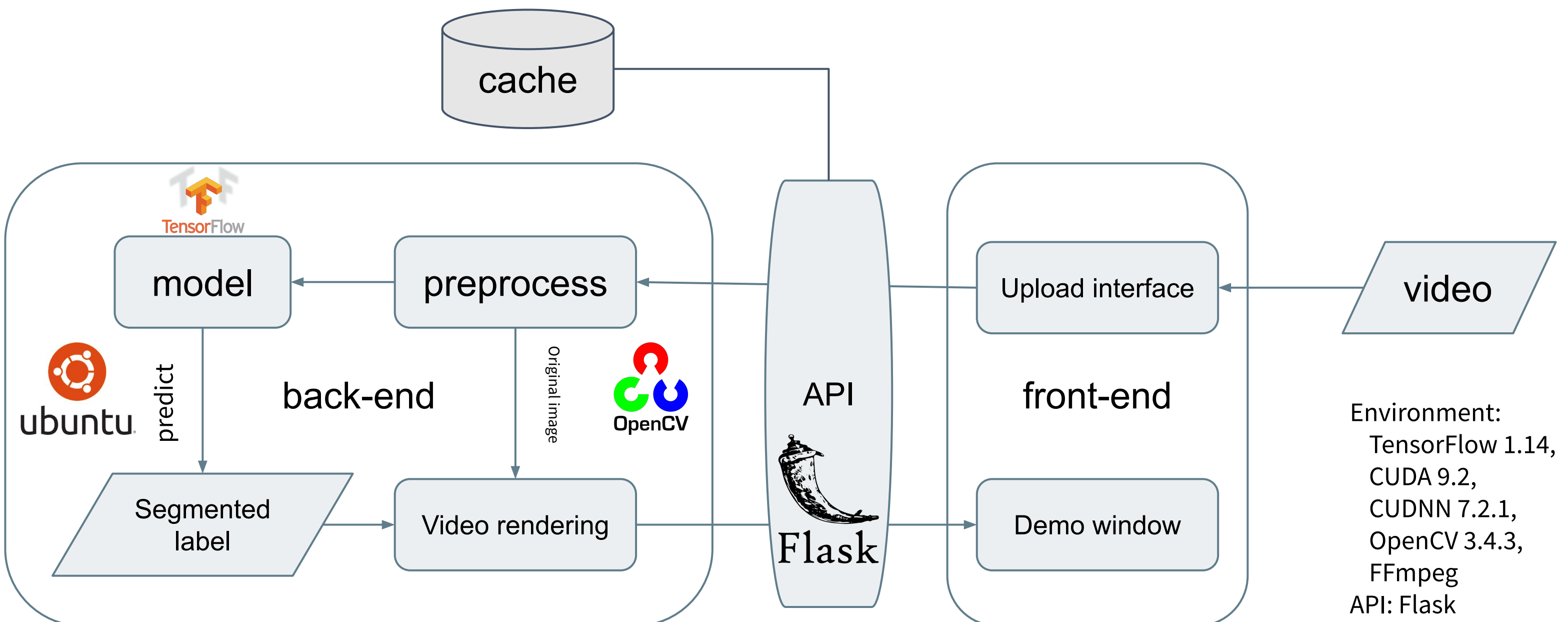
Model Evaluation

- Numerical evaluation:
  - Region similarity (avg): 76.4% paper: 79.8%
  - Contour accuracy(avg): 78.2% paper: 80.6%
  - Temporal stability(avg): 34.5 paper: 37.6



System Design

- Aims:
  - To support previewing uploaded video with segmented foreground object.
- Expected outcome:
  - Separate foreground objects with larger than 80% overlapping with ground-truth on average
  - Provide API for video website and simple web front-end for demo



Environment:
TensorFlow 1.14,
CUDA 9.2,
CUDNN 7.2.1,
OpenCV 3.4.3,
FFmpeg
API: Flask
Front-end:
django
video js/D3.js

Figure 5. Simple overview diagram of system

Web Interface

Project

- frontend ~/Documents/GitHub/ja
  - app
    - migrations
    - static
    - templates
      - gallery.html
      - homepage.html
      - upload.html
      - video.html
      - vis.html
    - __init__.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - views.py
  - frontend
    - __init__.py
    - settings.py
    - urls.py
    - wsgi.py
  - upload
    - File
    - First_Frame
  - .gitignore
  - db.sqlite3
  - manage.py

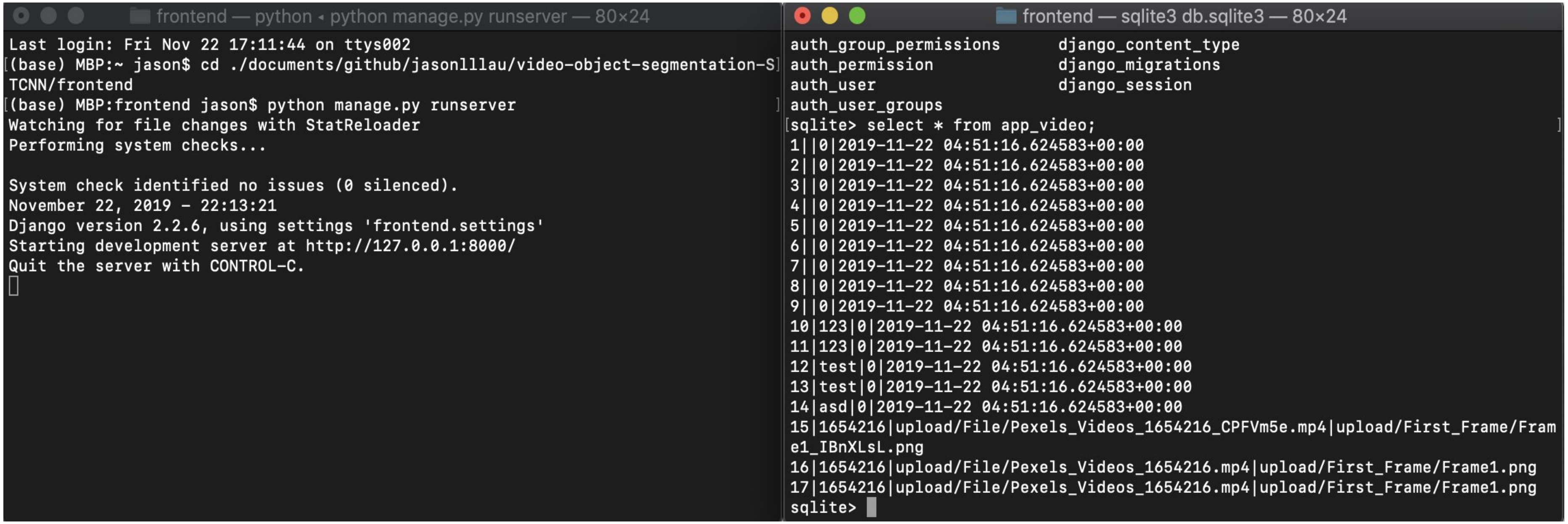
video.html | upload.html | homepage.html | views.py | vis.html | gallery.html

```python
url='http://127.0.0.1:5000/api/infer'

def homepage(request):

    return render_to_response('homepage.html')

def upload(request):
    if request.method == "POST":
        uf = VideoForm(request.POST, request.FILES)
        if uf.is_valid():  # if valid
            videoname = uf.cleaned_data['videoname']
            Frame = uf.cleaned_data['Frame']
            File = uf.cleaned_data['File']
            # save file
            video = Video()
            video.videoname = videoname
            video.Frame = Frame
            video.File = File
            result = video
            result.save()
            x = requests.post(url,data={"videoname":videoname},files=request.FILES)
            with open(os.path.join("app/static/video", videoname+".mp4"),"wb") as f:
                f.write(x.content)
            with open(os.path.join("app/static/headers", videoname+".json"),"w") as f:
                f.write(json.dumps(dict(x.headers)))
            v = os.path.join("/static/video", videoname+".mp4")
            h = os.path.join("/static/headers", videoname + ".json")
            x={"v":v,"h":h}
            return render_to_response('video.html',x)
    else:
        uf = VideoForm()

    return render_to_response('upload.html',{'uf':uf})


def video(request):
    return render_to_response('video.html')
```

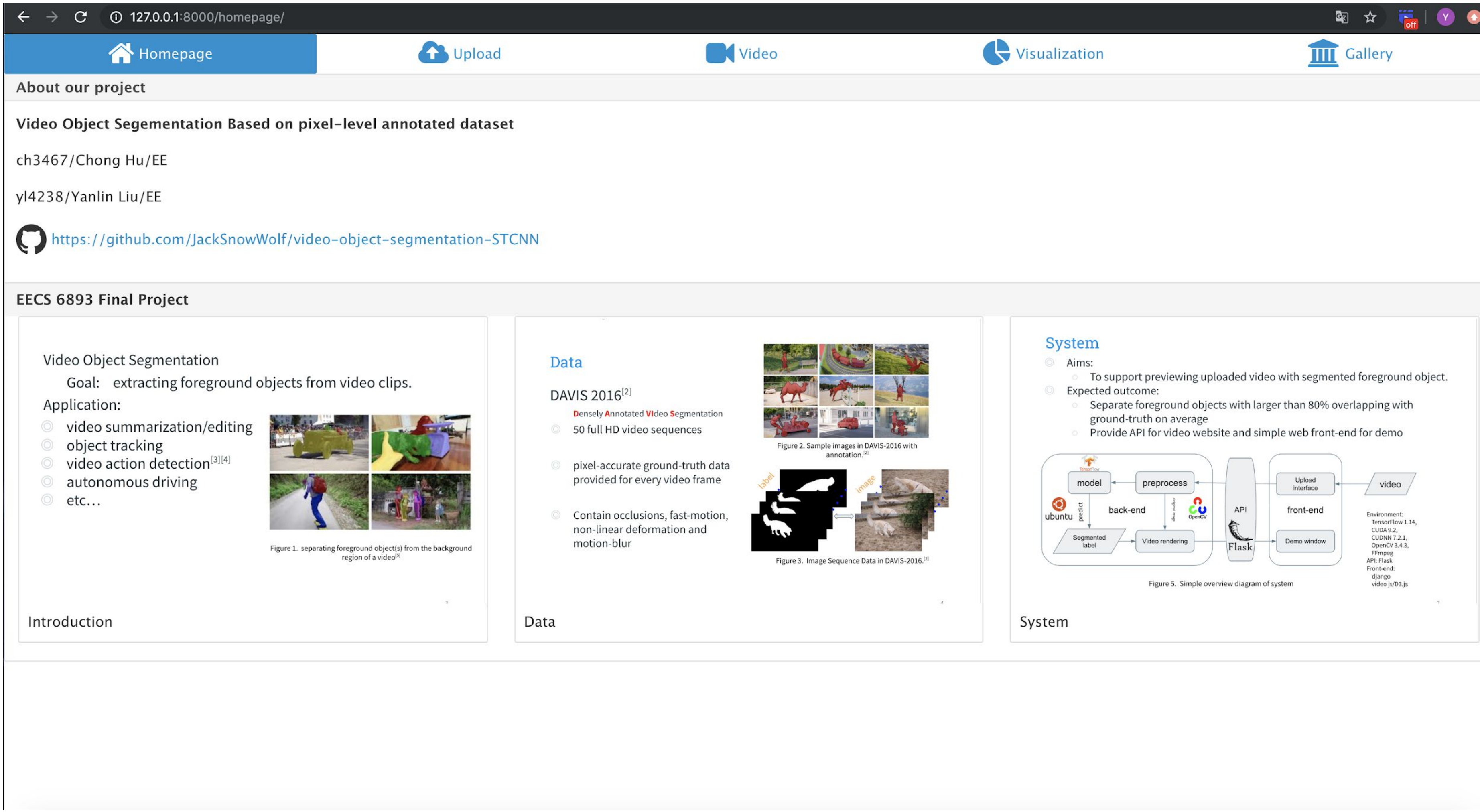
video.html | upload.html | homepage.html | views.py | app.py | vis.html

```python
@app.route('/api/infer', methods=['GET', 'POST'])
def infer():
    try:
        logging.info("receive video and analysis original video")
        if request.method == 'POST':
            video_file = request.files.get("File")
            first_mask = request.files.get("Frame")
            video_name = request.form.get("videoname")
            video_name = os.path.splitext(video_name)[0]
            video_path, img_path, first_mask_path = video_utils.init_video(
                video_name)

            video_name_res = video_name + "-concat.mp4"
            if video_name_res in video_utils.cached_video:
                logging.info("Hit video in the cache")
                return send_file(
                    os.path.join(video_path).replace(".mp4", "-concat.mp4"))

            video_file.save(video_path)
            first_mask.save(first_mask_path)

            tag = video_name not in video_utils.cached_model
            if tag:
                logging.info("Hit previous model weight")
            else:
                logging.info("finetune pre-trained model for new sequence")
            video_utils.video2img(video_path, img_path)
            osvos_demo.demo(seq_name=video_name, first_mask=first_mask_path,
                            img_path=img_path,
                            result_path=os.path.join('tmp', video_name, 'pred'),
                            concate_path=os.path.join('tmp', video_name,
                                                      'concat'),
                            train_model=tag)
            result_video_path = video_utils.render_video(video_name)

            return send_file(result_video_path)
        else:
            raise ValueError("Not support GET method")
    except ValueError or AssertionError as e:
        logging.error(e)
        return "Internal error.", 504, {
            'Content-Type': 'text/plain; charset=utf-8'}
```

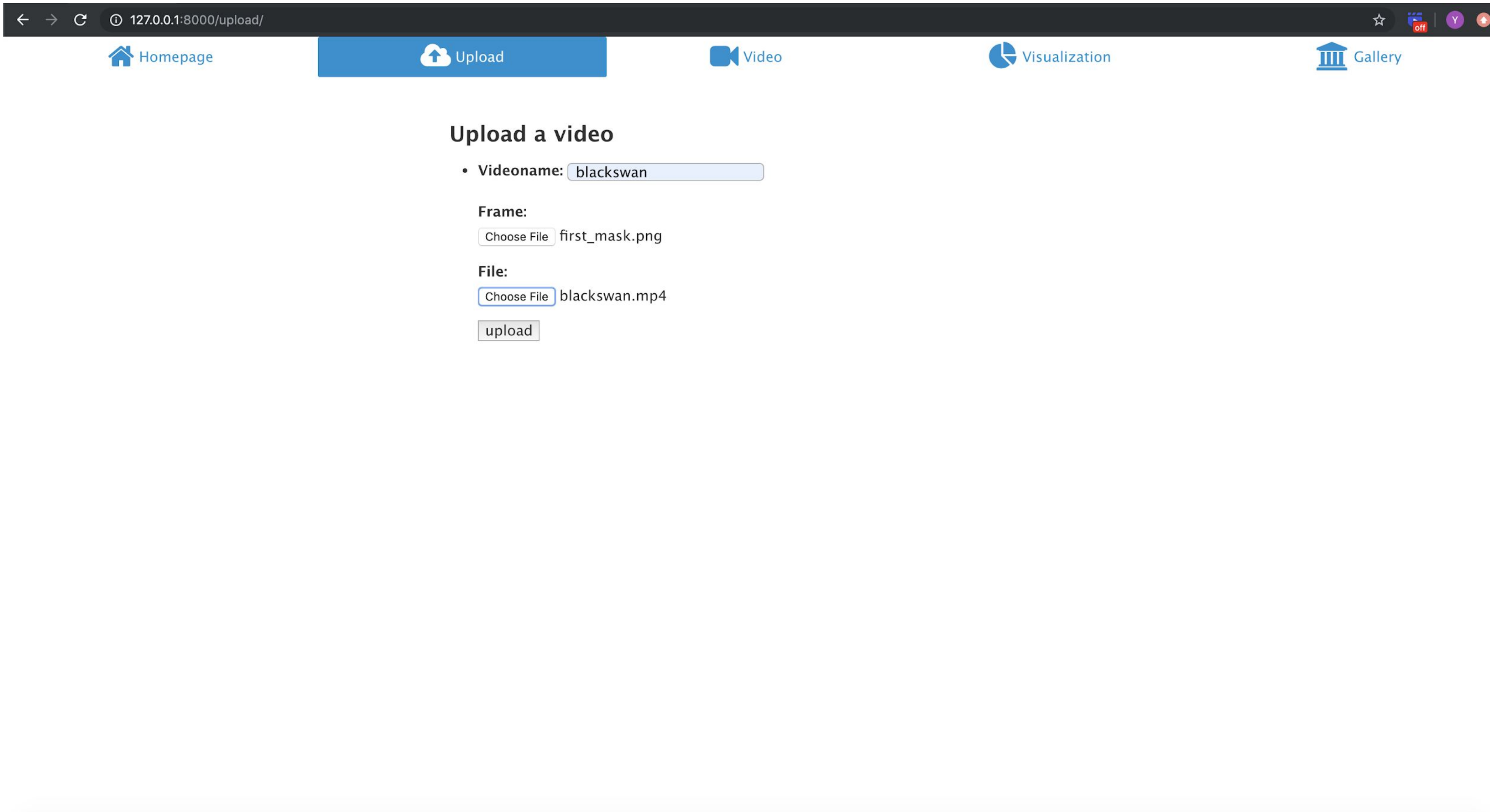
Web Interface



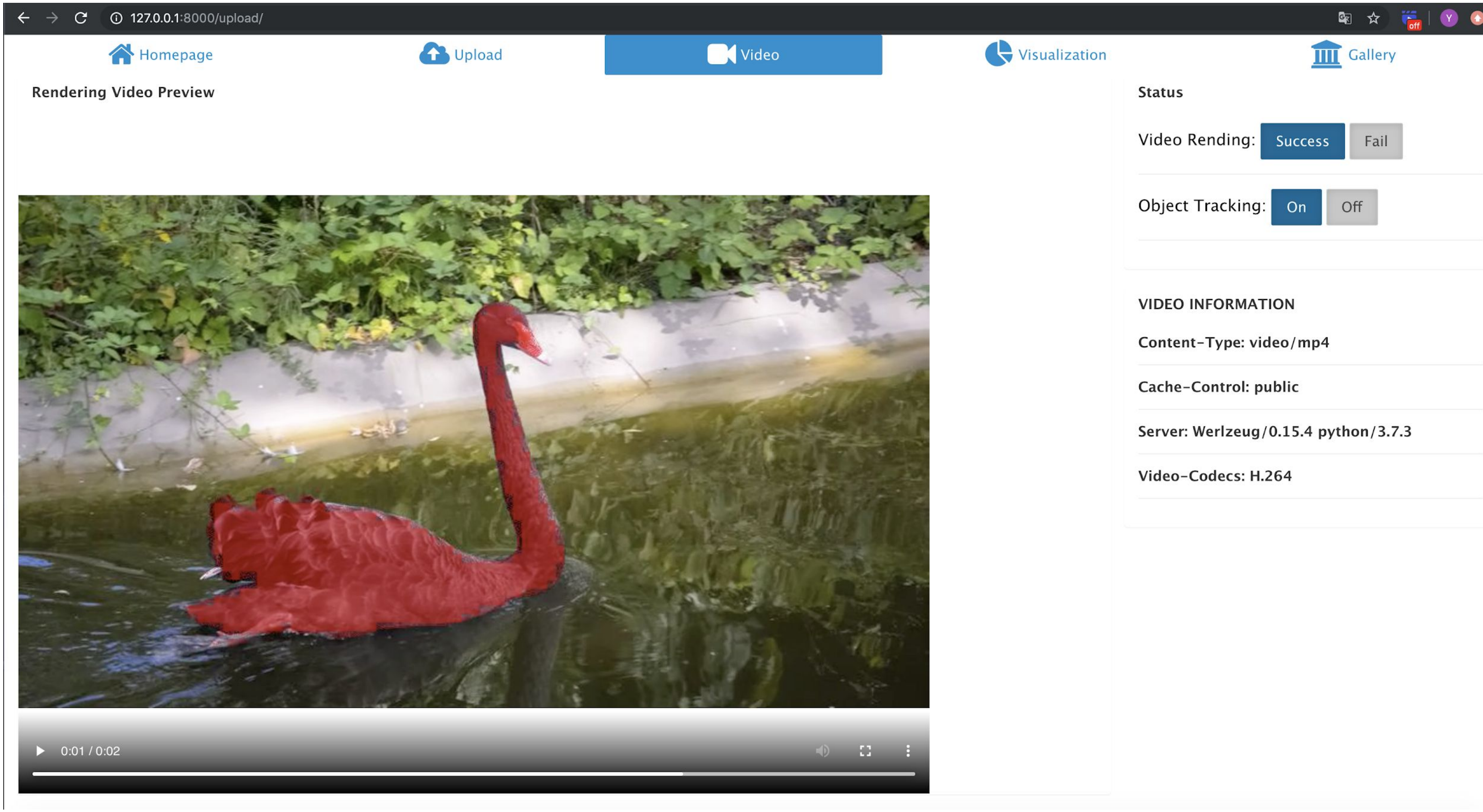
Web Interface



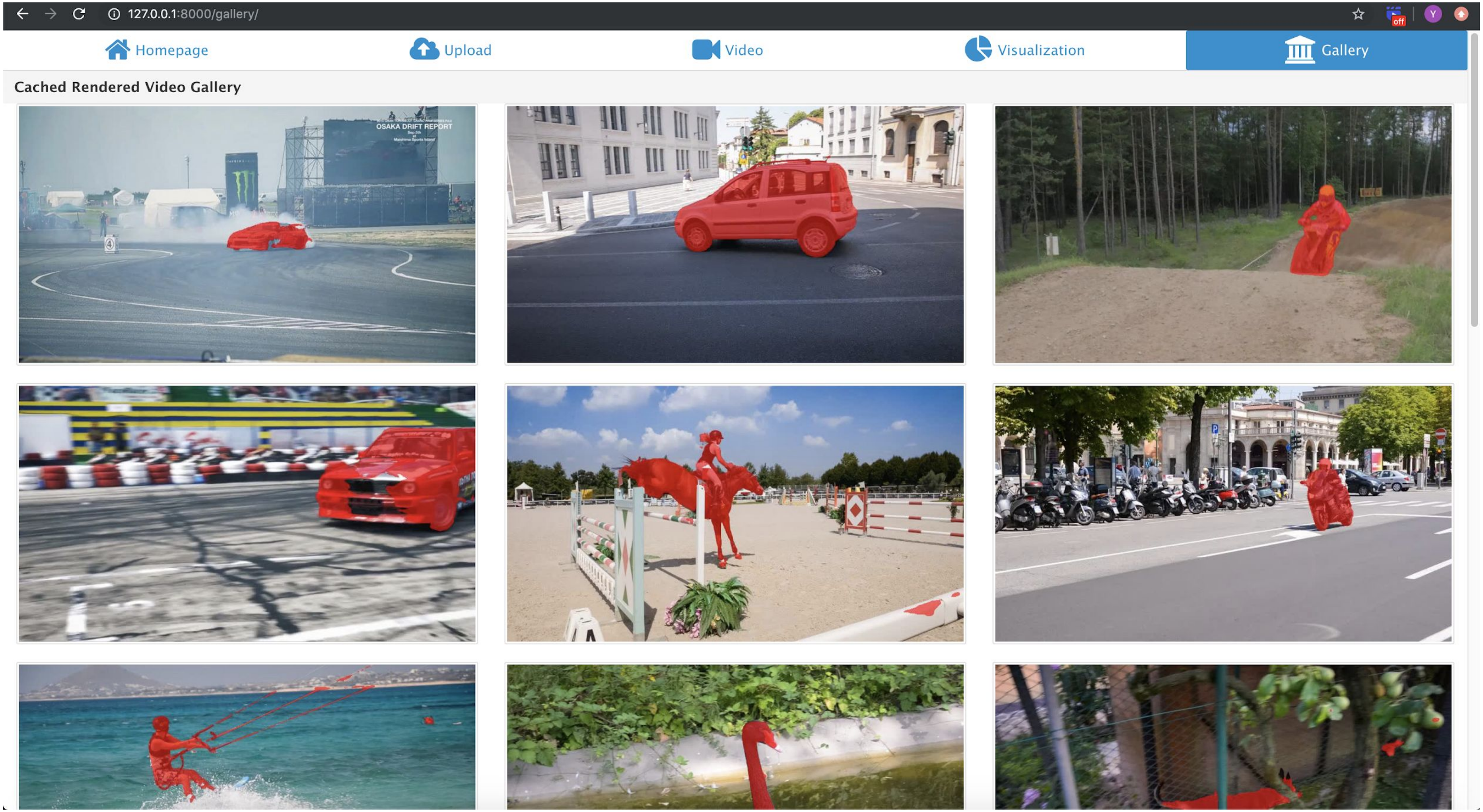
Web Interface



Web Interface



Web Interface



Visualization

Data:

Obtain the position information (x, y) of the target object in each frame of the rendered video, including the midpoint position, upper left corner position, upper right corner position, lower left corner position, and lower right corner position of the object's rectangular frame.

goat.txt

216.133242	451.197652	71.000000	349.000000	389.000000	565.000000
217.879220	460.034676	72.000000	353.000000	394.000000	572.000000
222.185448	466.956246	77.000000	354.000000	400.000000	577.000000
229.893188	472.707613	84.000000	356.000000	409.000000	588.000000
242.092474	478.618579	94.000000	359.000000	421.000000	591.000000
246.269106	485.419715	102.000000	369.000000	419.000000	601.000000
242.950754	490.974462	100.000000	381.000000	409.000000	600.000000
242.804330	494.749247	98.000000	391.000000	412.000000	600.000000
245.935857	502.061572	101.000000	400.000000	415.000000	607.000000
244.924871	510.883006	104.000000	412.000000	418.000000	616.000000
244.903987	512.333416	106.000000	417.000000	419.000000	615.000000
242.092352	513.830122	106.000000	413.000000	418.000000	624.000000
240.435691	522.427074	105.000000	419.000000	419.000000	628.000000
241.280114	536.318080	109.000000	432.000000	421.000000	639.000000
245.479277	544.397718	119.000000	433.000000	423.000000	651.000000
249.589315	546.057948	133.000000	430.000000	426.000000	662.000000
252.448494	548.882142	146.000000	433.000000	431.000000	670.000000
250.791711	556.397977	150.000000	440.000000	428.000000	678.000000
251.163898	564.106555	154.000000	450.000000	422.000000	686.000000
251.322119	571.199169	153.000000	464.000000	412.000000	700.000000
252.772047	580.763927	157.000000	476.000000	398.000000	711.000000
253.901185	590.498668	159.000000	484.000000	380.000000	720.000000
244.602829	595.192276	150.000000	490.000000	354.000000	728.000000
232.764329	602.469868	140.000000	496.000000	345.000000	740.000000
226.913854	612.481714	137.000000	504.000000	344.000000	748.000000
221.082112	617.410812	137.000000	508.000000	343.000000	753.000000
221.757810	611.296934	137.000000	499.000000	346.000000	746.000000
223.968242	606.710576	138.000000	494.000000	348.000000	743.000000
219.295651	611.330340	132.000000	494.000000	345.000000	747.000000
217.172108	612.290865	126.000000	491.000000	345.000000	749.000000

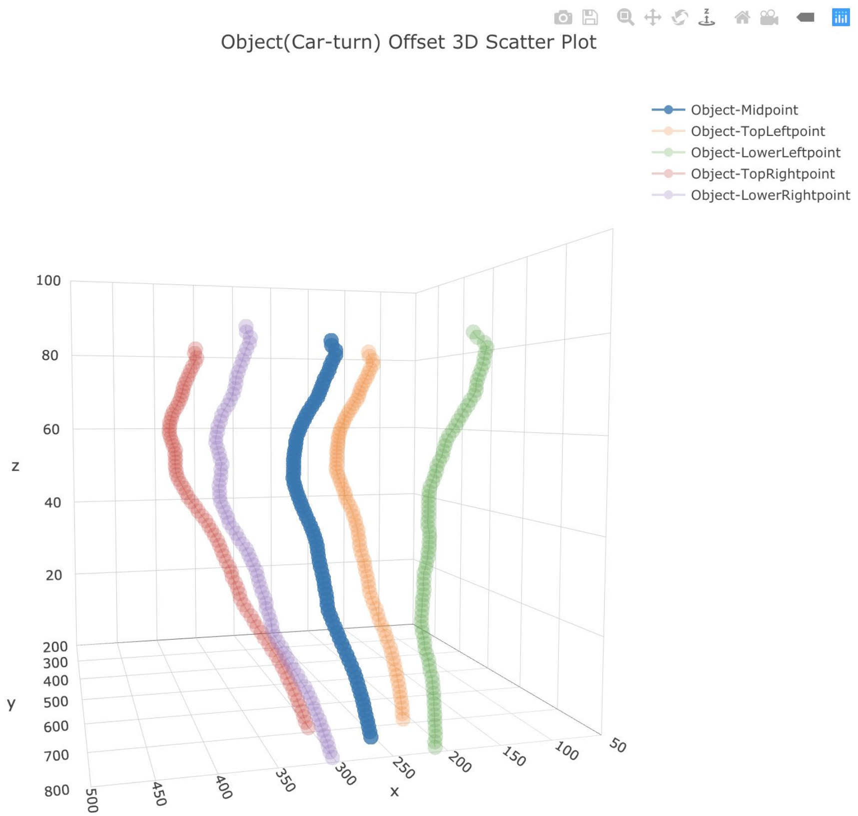
3D scatter plot:

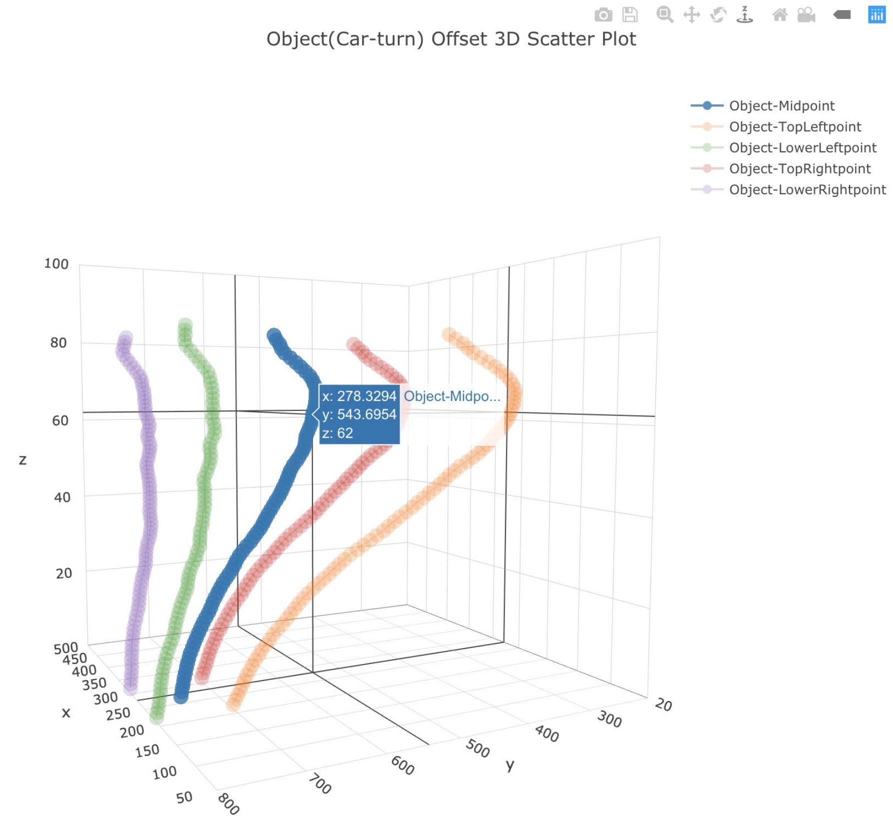
- Describe the overall offset of the position coordinates of the object we are tracking, corresponding to the motion trajectory of the object in the video.
- x-axis and y-axis represent the horizontal and vertical coordinates of the object.
- z-axis represents time, in unit of each frame

Visualization

First Video: Car-turn.mp4
Relatively steady object motion trajectory

Visualization

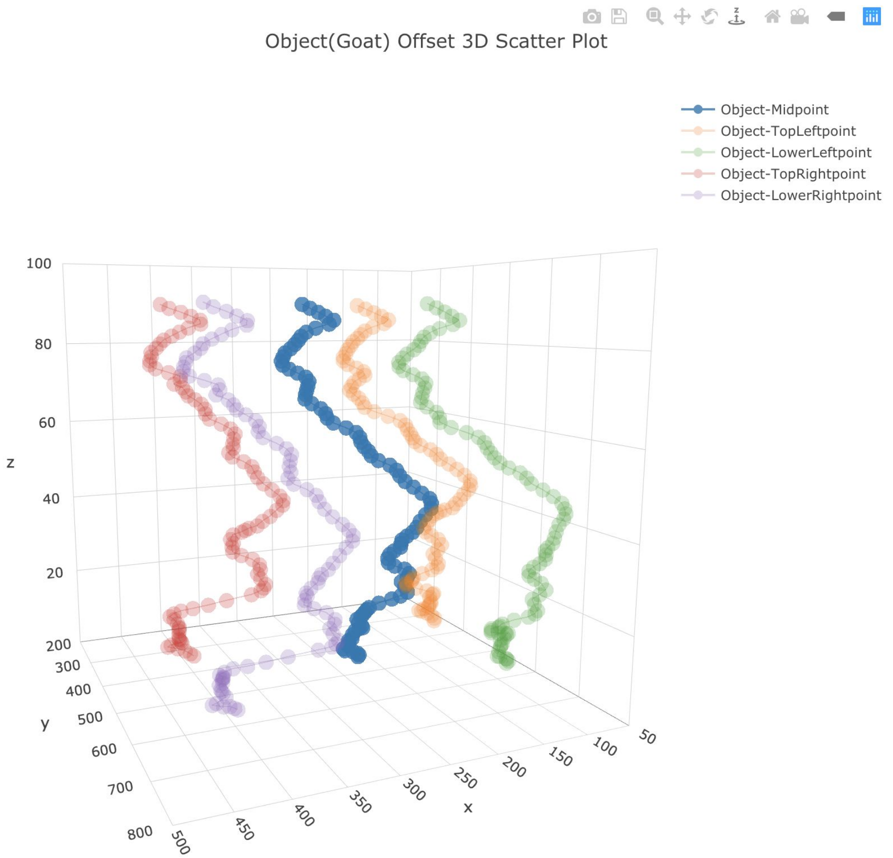


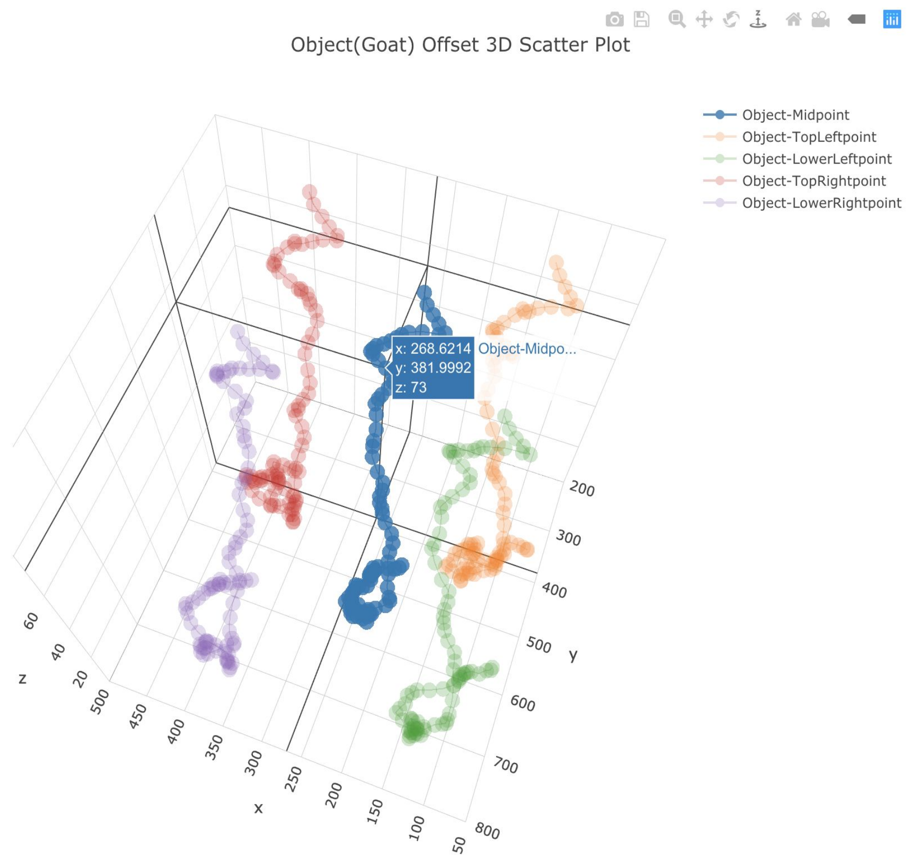


Visualization

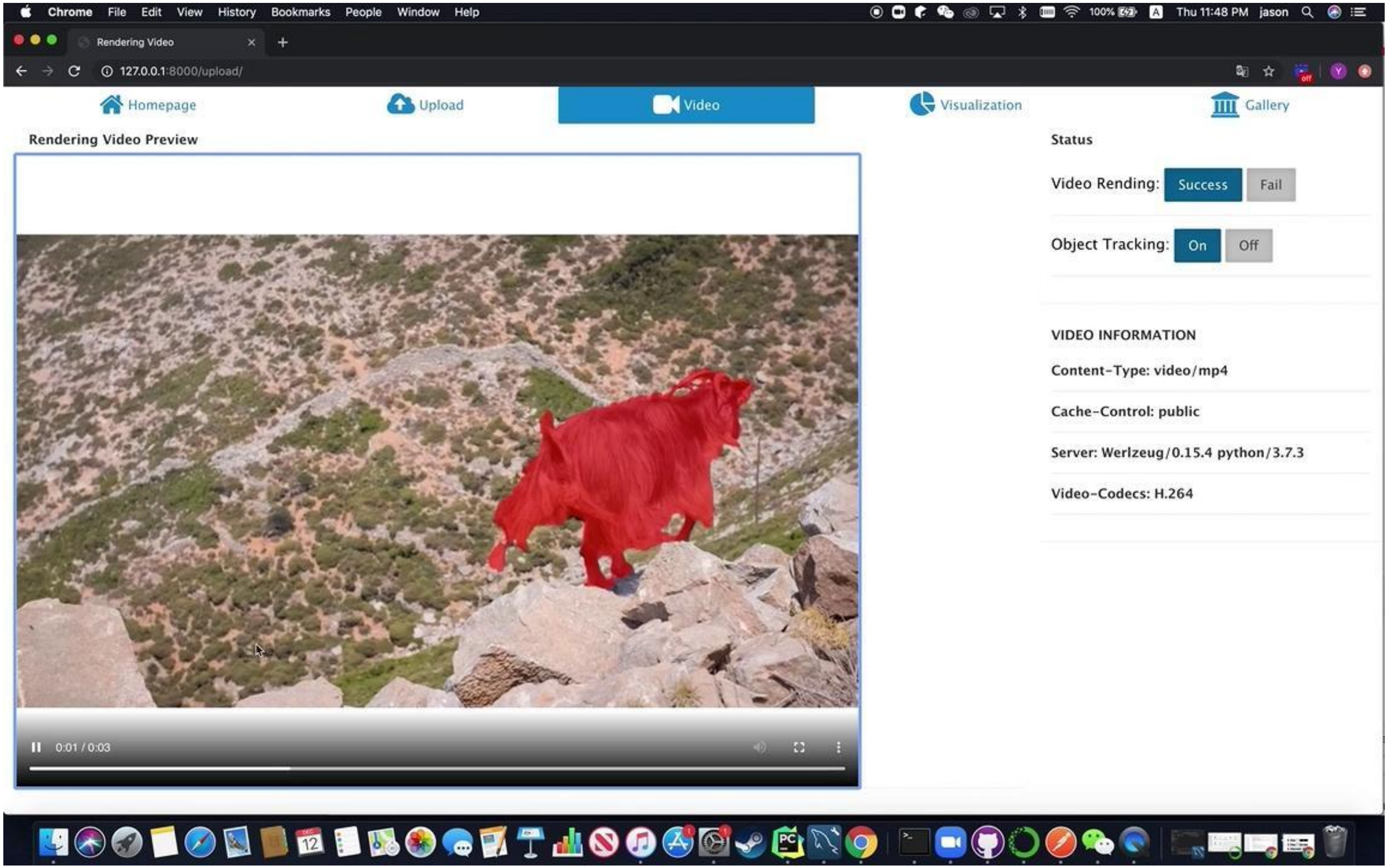
Second Video: Goat.mp4
Relatively unstable object motion trajectory

Visualization





Final Results



Reference

[1] Kai Xu, Longyin Wen, Guorong Li, Liefeng Bo, and Qingming Huang. Spatiotemporal cnn for video object segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[2] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[3] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A. Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. CoRR, abs/1705.08421, 2017
[4] Xitong Yang, Xiaodong Yang, Ming-Yu Liu, Fanyi Xiao, Larry S. Davis, and Jan Kautz. Step: Spatio-temporal progressive learning for video action detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[5] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbel´aez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. arXiv:1704.00675, 2017
[6] Fanyi Xiao and Yong Jae Lee. Track and segment: An iterative unsupervised approach for video object proposals. In CVPR, 2016. 1, 2
[7] Nicolas Marki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral space video segmentation. In CVPR, pages 743–751, 2016. 6, 8
[8] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstraint video. In ICCV, pages 1–6, 2013. 2
[9] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J. Black. Video segmentation via object flow. In CVPR, pages 3899–3908, 2016. 1, 2, 6, 8
[10] Longyin Wen, Dawei Du, Zhen Lei, Stan Z. Li, and Ming-Hsuan Yang. JOTS: joint online tracking and segmentation. In CVPR, pages 2226–2234, 2015. 1, 2
[11] Siyang Li, Bryan Seybold, Alexey Vorobyov, Xuejing Lei, and C.-C. Jay Kuo. Unsupervised video object segmentation with motion-based bilateral networks. In ECCV, pages 215– 231, 2018. 1

Thank you !