

# 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***

1. Introduction
2. Data Preprocess
3. Model
4. System
5. Web Interface
6. Data Visualization
7. Final Results

# ***Introduction***

## Video Object Segmentation

Goal: extracting foreground objects from video clips.

Application:

- ◎ video summarization/editing
- ◎ object tracking
- ◎ video action detection<sup>[3][4]</sup>
- ◎ autonomous driving
- ◎ etc...



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

# Data Preprocess

## Data

### DAVIS 2016<sup>[2]</sup>

- Densely Annotated VIdeo Segmentaion
- ◎ 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.<sup>[2]</sup>

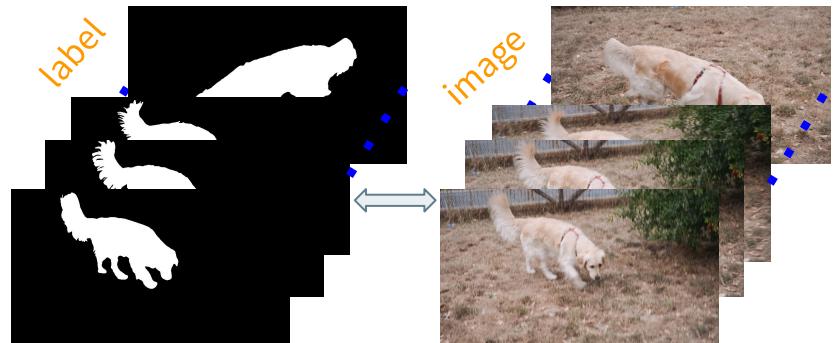
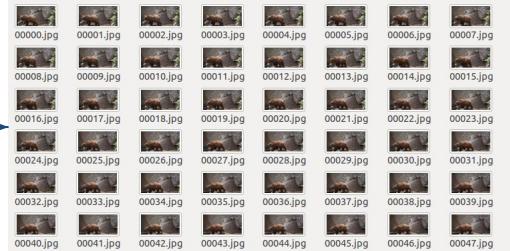
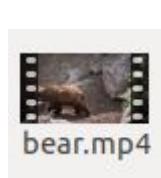


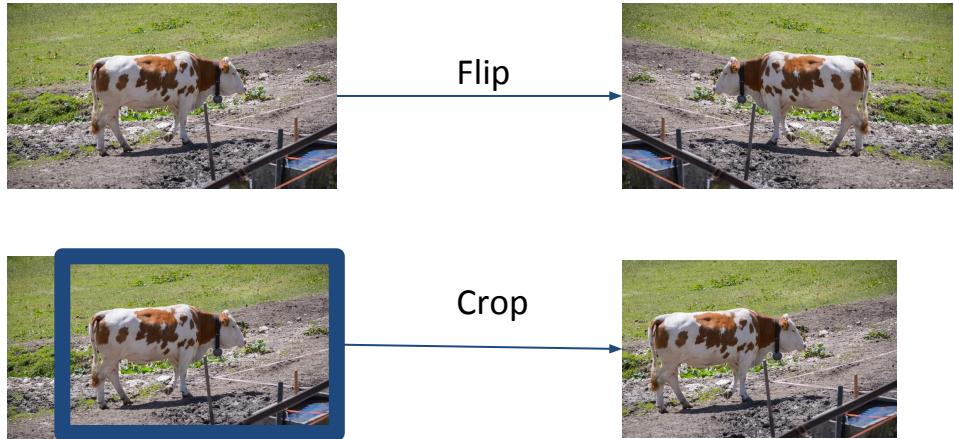
Figure 3. Image Sequence Data in DAVIS-2016.<sup>[2]</sup>

# 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

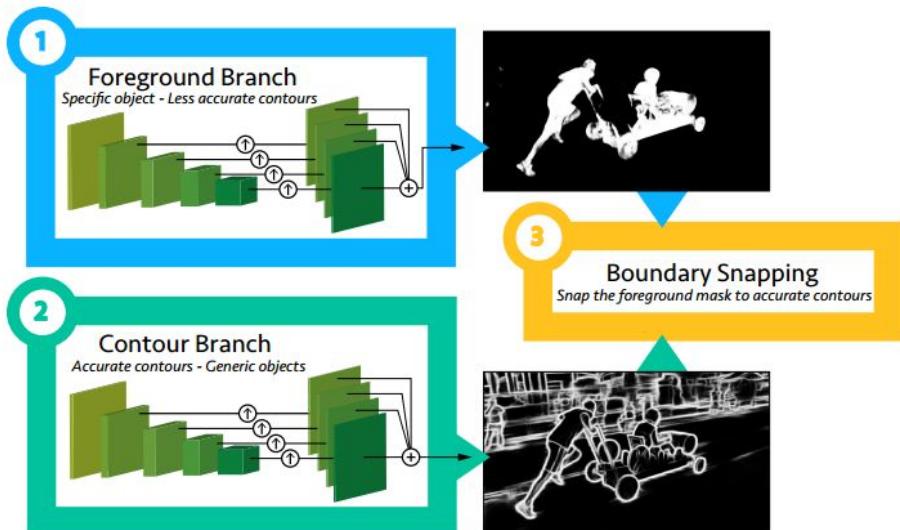


# Model Construction

- 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

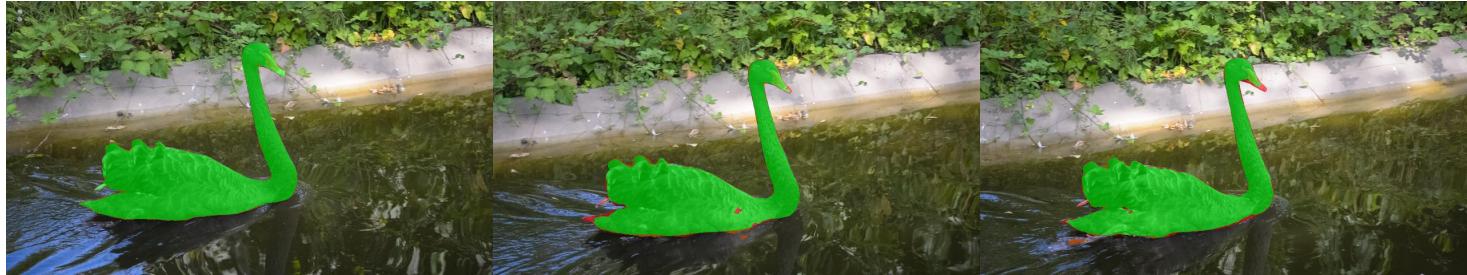
$$\mathcal{L}_{mod} = -\beta \sum_{j \in Y_+} \log P(y_j=1|X) - (1-\beta) \sum_{j \in Y_-} \log P(y_j=0|X) \quad (1)$$

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

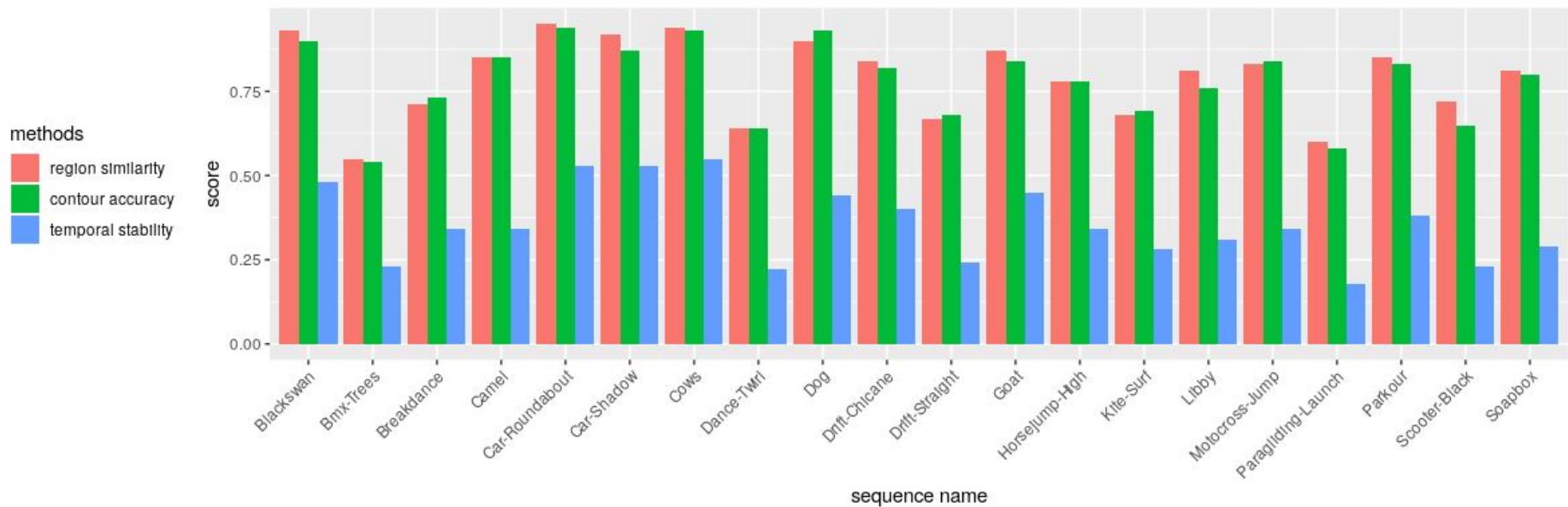
frame 38

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

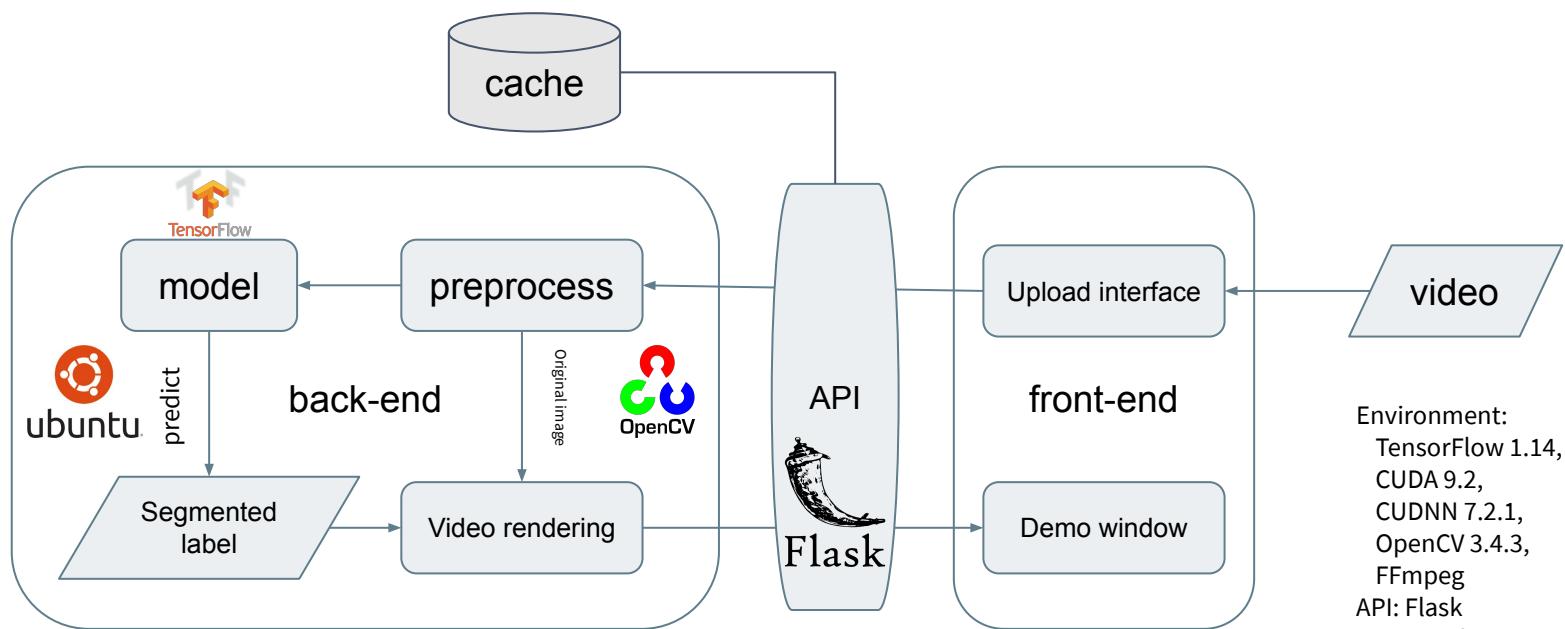


Figure 5. Simple overview diagram of system

# Web Interface

The screenshot shows a code editor with several tabs open, displaying Python code for a web application. The project structure on the left includes files like `app.py`, `migrations`, `static`, `templates` (containing `gallery.html`, `homepage.html`, `upload.html`, `video.html`, `vis.html`), `__init__.py`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `views.py`, `frontend` (containing `__init__.py`, `settings.py`, `urls.py`, `wsgi.py`), `upload` (containing `File`, `First_Frame`), `.gitignore`, `db.sqlite3`, and `manage.py`.

The code in the tabs includes:

- `views.py`: Handles homepage, upload, and video requests.
- `app.py`: Handles API routes for inference and video processing.

```
Project ~/Documents/GitHub/ja
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 ×
video.html × upload.html × homepage.html × views.py × app.py × vis.html ×

14 url='http://127.0.0.1:5000/api/infer'
15
16 def homepage(request):
17
18     return render_to_response('homepage.html')
19
20 def upload(request):
21     if request.method == "POST":
22         uf = VideoForm(request.POST, request.FILES)
23         if uf.is_valid(): # if valid
24             videoname = uf.cleaned_data['videoname']
25             Frame = uf.cleaned_data['Frame']
26             File = uf.cleaned_data['File']
27             # save file
28             video = Video()
29             video.videoname = videoname
30             video.Frame = Frame
31             video.File = File
32             result = video
33             result.save()
34             x = requests.post(url,data={"videoname":videoname},files=request.FILES)
35             with open(os.path.join("app/static/video", videoname+".mp4"),"wb") as f:
36                 f.write(x.content)
37                 with open(os.path.join("app/static/headers", videoname+".json"),"w") as f:
38                     f.write(json.dumps(dict(x.headers)))
39                     v = os.path.join("static/video", videoname+".mp4")
40                     h = os.path.join("static/headers", videoname + ".json")
41                     x={"v":v,"h":h}
42                     return render_to_response('video.html',x)
43             else:
44                 uf = VideoForm()
45
46             return render_to_response('upload.html',{'uf':uf})
47
48 def video(request):
49     return render_to_response('video.html')
50
51 def visit(request):
52
53 @app.route('/api/infer', methods=['GET', 'POST'])
54 def infer():
55     try:
56         logging.info("receive video and analysis original video")
57         if request.method == 'POST':
58             video_file = request.files.get("File")
59             first_mask = request.files.get("Frame")
60             video_name = request.form.get("videoname")
61             video_name = os.path.splitext(video_name)[0]
62             video_path, img_path, first_mask_path = video_utils.init_video(
63                 video_name)
64
65             video_name_res = video_name + "-concat.mp4"
66             if video_name_res in video_utils.cached_video:
67                 logging.info("Hit video in the cache")
68                 return send_file(
69                     os.path.join(video_path).replace(".mp4", "-concat.mp4"))
70
71             video_file.save(video_path)
72             first_mask.save(first_mask_path)
73
74             tag = video_name not in video_utils.cached_model
75             if tag:
76                 logging.info("Hit previous model weight")
77             else:
78                 logging.info("finetune pre-trained model for new sequence")
79                 video_utils.video2img(video_path, img_path)
80                 osvos_demo.demo(seq_name=video_name, first_mask=first_mask_path,
81                                 img_path=img_path,
82                                 result_path=os.path.join('tmp', video_name, 'pred'),
83                                 concate_path=os.path.join('tmp', video_name,
84                               'concat'),
85                                 train_model=tag)
86                 result_video_path = video_utils.render_video(video_name)
87
88                 return send_file(result_video_path)
89             else:
90                 raise ValueError("Not support GET method")
91             except ValueError or AssertionError as e:
92                 logging.error(e)
93                 return "Internal error.", 504, {
94                   'Content-Type': 'text/plain; charset=utf-8'}
```

# Web Interface

The image shows two terminal windows side-by-side. The left window, titled 'frontend — python manage.py runserver — 80x24', displays the output of a Django development server. It includes the system check results, the Django version ('2.2.6'), the server address ('http://127.0.0.1:8000/'), and instructions to quit ('Quit the server with CONTROL-C'). The right window, titled 'frontend — sqlite3 db.sqlite3 — 80x24', shows the results of a SQL query on an SQLite database named 'db.sqlite3'. The query selects all columns from the 'app\_video' table. The results are listed as rows of data, each consisting of a timestamp (e.g., '2019-11-22 04:51:16.624583+00:00') and a file path ('upload/File/Pexels\_Videos\_1654216\_CPFVm5e.mp4').

```
Last login: Fri Nov 22 17:11:44 on ttys002
[(base) MBP:~ jason$ cd ./documents/github/jasonllau/video-object-segmentation-S
TCNN/frontend
[(base) MBP:frontend jason$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
November 22, 2019 - 22:13:21
Django version 2.2.6, using settings 'frontend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

auth_group_permissions      django_content_type
auth_permission             django_migrations
auth_user                   django_session
auth_user_groups
[sqlite> select * from app_video;
1||0|2019-11-22 04:51:16.624583+00:00
2||0|2019-11-22 04:51:16.624583+00:00
3||0|2019-11-22 04:51:16.624583+00:00
4||0|2019-11-22 04:51:16.624583+00:00
5||0|2019-11-22 04:51:16.624583+00:00
6||0|2019-11-22 04:51:16.624583+00:00
7||0|2019-11-22 04:51:16.624583+00:00
8||0|2019-11-22 04:51:16.624583+00:00
9||0|2019-11-22 04:51:16.624583+00:00
10|123|0|2019-11-22 04:51:16.624583+00:00
11|123|0|2019-11-22 04:51:16.624583+00:00
12|test|0|2019-11-22 04:51:16.624583+00:00
13|test|0|2019-11-22 04:51:16.624583+00:00
14|asd|0|2019-11-22 04:51:16.624583+00:00
15|1654216|upload/File/Pexels_Videos_1654216_CPFVm5e.mp4|upload/First_Frame/Fram
e1_IBnXlsL.png
16|1654216|upload/File/Pexels_Videos_1654216.mp4|upload/First_Frame/Frame1.png
17|1654216|upload/File/Pexels_Videos_1654216.mp4|upload/First_Frame/Frame1.png
sqlite>
```

# Web Interface

127.0.0.1:8000/homepage/

Homepage Upload Video Visualization Gallery

About our project

Video Object Segmentation Based on pixel-level annotated dataset

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

 <https://github.com/JackSnowWolf/video-object-segmentation-STCNN>

EECS 6893 Final Project

Video Object Segmentation  
Goal: extracting foreground objects from video clips.  
Application:

- video summarization/editing
- object tracking
- video action detection<sup>[3][4]</sup>
- autonomous driving
- etc...

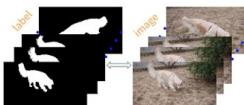
  
Figure 1. separating foreground object(s) from the background region of a video<sup>[3]</sup>

Data

DAVIS 2016<sup>[2]</sup>  
 Densely Annotated Video Segmentation  

- 50 full HD video sequences
- 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.<sup>[2]</sup>

  
Figure 3. Image Sequence Data in DAVIS-2016.<sup>[2]</sup>

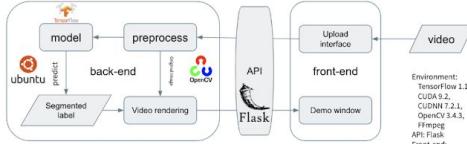
System

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 9.0,  
cuDNN 7.2.1,  
OpenCV 3.4.3,  
Python 3.6.5,  
API: Flask  
Front-end:  
django  
video.js/3.js

Introduction

Figure 5. Simple overview diagram of system

# Web Interface

← → ⌂ ① 127.0.0.1:8000/upload/ ⭐ off | Y ⚙

Homepage Upload Video Visualization Gallery

### Upload a video

- Videoname:

Frame:

Choose File first\_mask.png

File:

Choose File blackswan.mp4

upload

# Web Interface

127.0.0.1:8000/upload/

Homepage      Upload      **Video**      Visualization      Gallery

Rendering Video Preview



0:01 / 0:02

Status

Video Rending: Success Fail

Object Tracking: On Off

VIDEO INFORMATION

Content-Type: video/mp4

Cache-Control: public

Server: Werkzeug/0.15.4 python/3.7.3

Video-Codecs: H.264

# Web Interface

← → ⌂ 127.0.0.1:8000/gallery/ ⭐ 🔍 Y 🎯

Homepage Upload Video Visualization Gallery

Cached Rendered Video Gallery

The web interface displays a collection of 12 images arranged in a 4x3 grid. Each image features a red 3D object or character superimposed onto a real-world scene. The objects include a red car performing a drift, a red SUV driving on a city street, a person in a red suit walking through a forest, a red truck on a racing track, a person riding a horse at an equestrian competition, a red figure standing next to a motorcycle, a red figure sitting on a beach, a red figure windsurfing, a red swan statue in a garden, and a red deer statue behind a fence. The interface has a dark header with navigation icons and a blue footer bar.

# 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
244.9003987	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

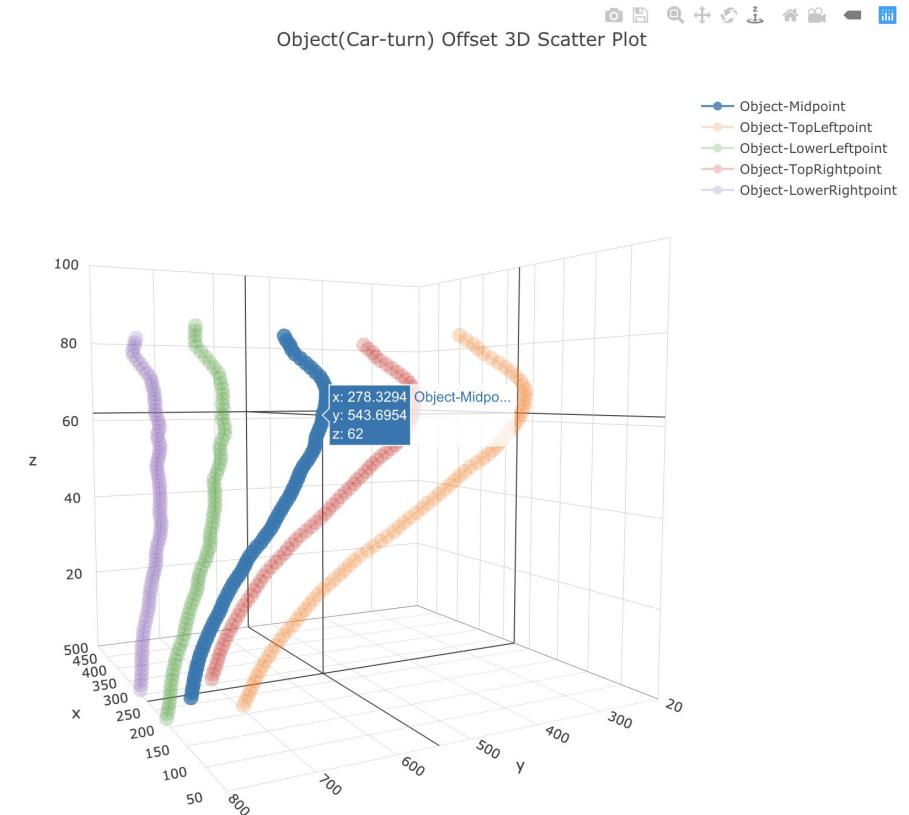
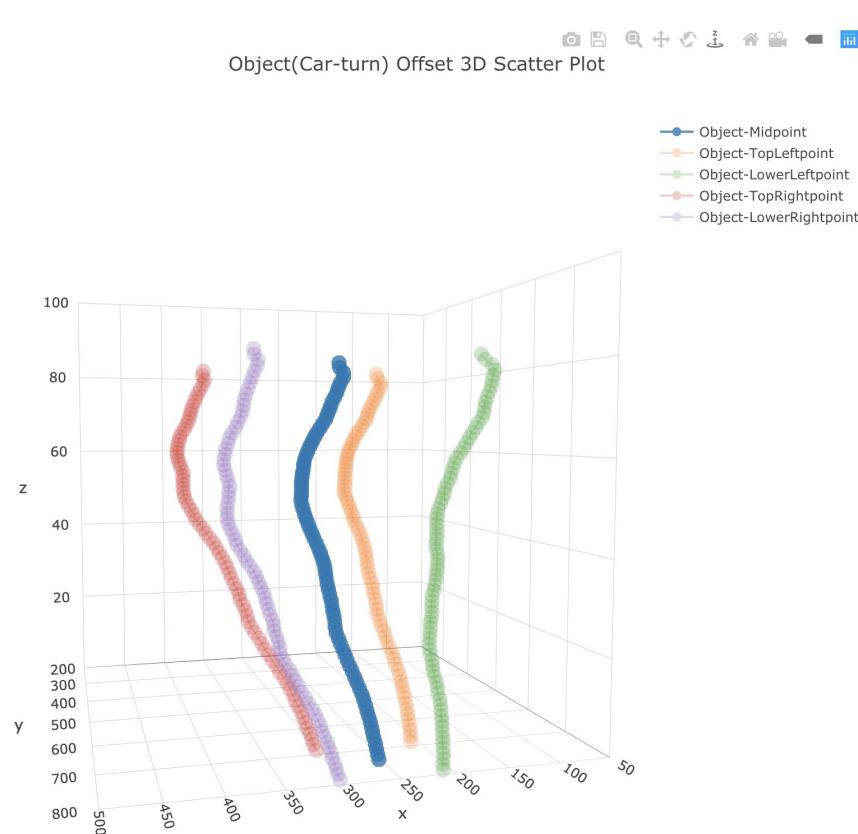
# *Visualization*

First Video: Car-turn.mp4

Relatively steady object motion trajectory



# Visualization



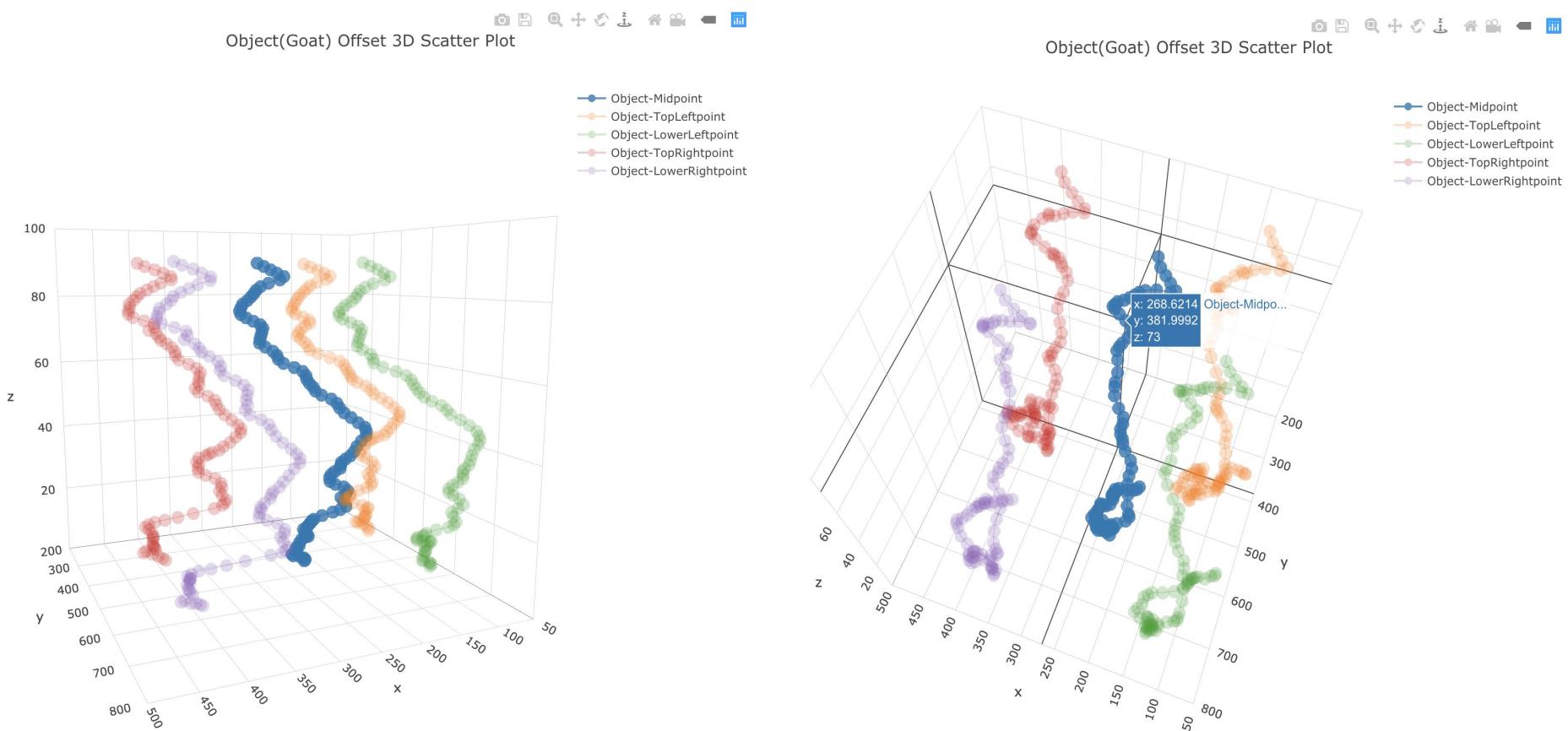
# *Visualization*

Second Video: Goat.mp4

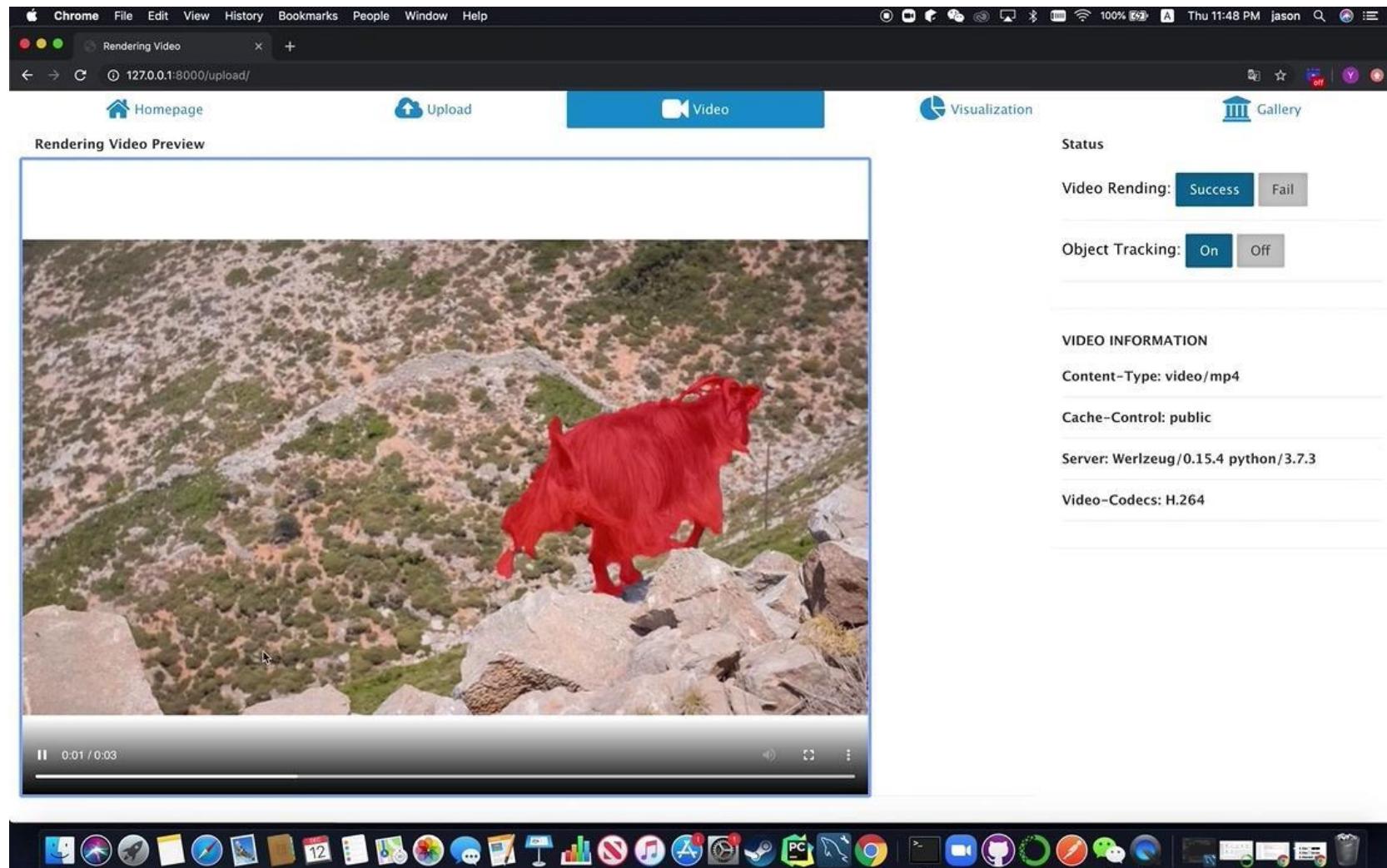
Relatively unstable object motion trajectory



# 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 Arbelaez, 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 !