Feature Extraction with OpenFace

In [7]:
```
from tqdm import tqdm
```

In [8]:
```
#setting up the detector
from feat import Detector

detector = Detector()
detector
```

Out[8]:
```
feat.detector.Detector(face_model=retinaface, landmark_model=mobilefacenet, au_model=xgb,
emotion_model=resmasknet, facepose_model=img2pose)
```

In [13]:
```
#testing the output for a single video
test_video_path = "C:\\Work\\606Capstone\\Video_chunks\\Video_chunks\\trial_lie_001_000.mp
video_prediction = detector.detect_video(test_video_path, skip_frames=24)
video_prediction.head()
```

```
100%|██████████████████████████████████████████████████████████████████████████████|
6/6 [00:27<00:00,  4.58s/it]
```

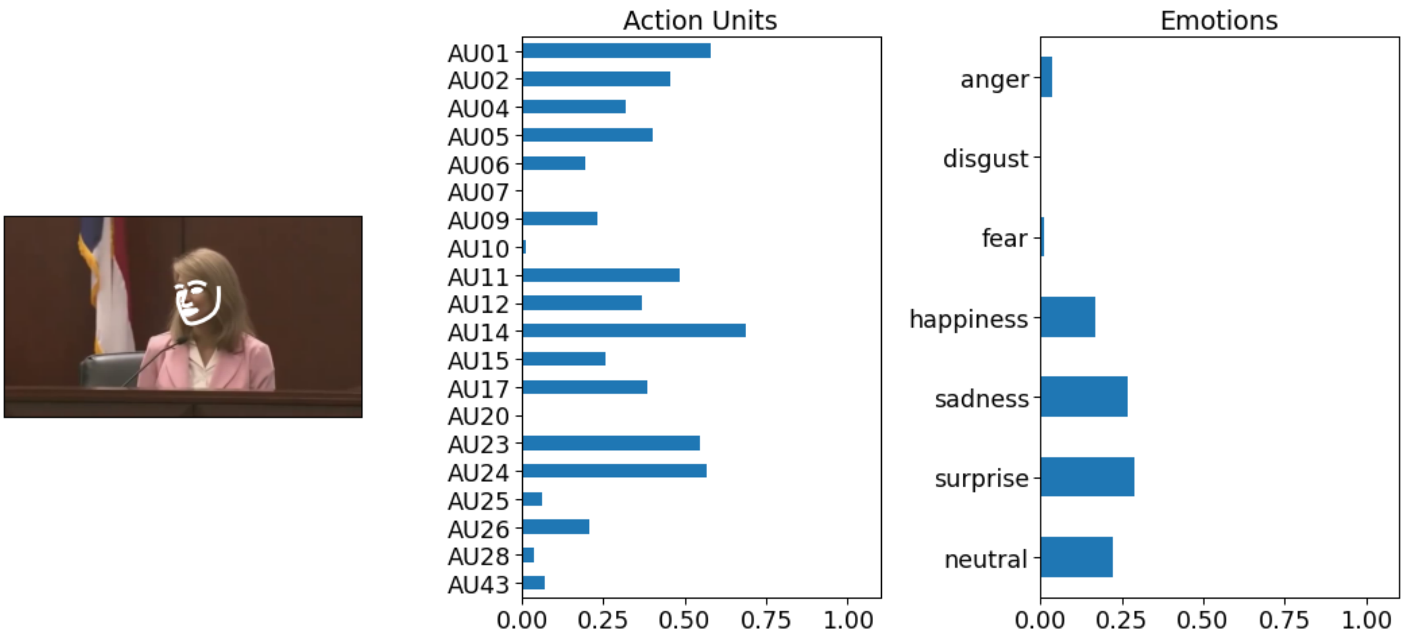
Out[13]:

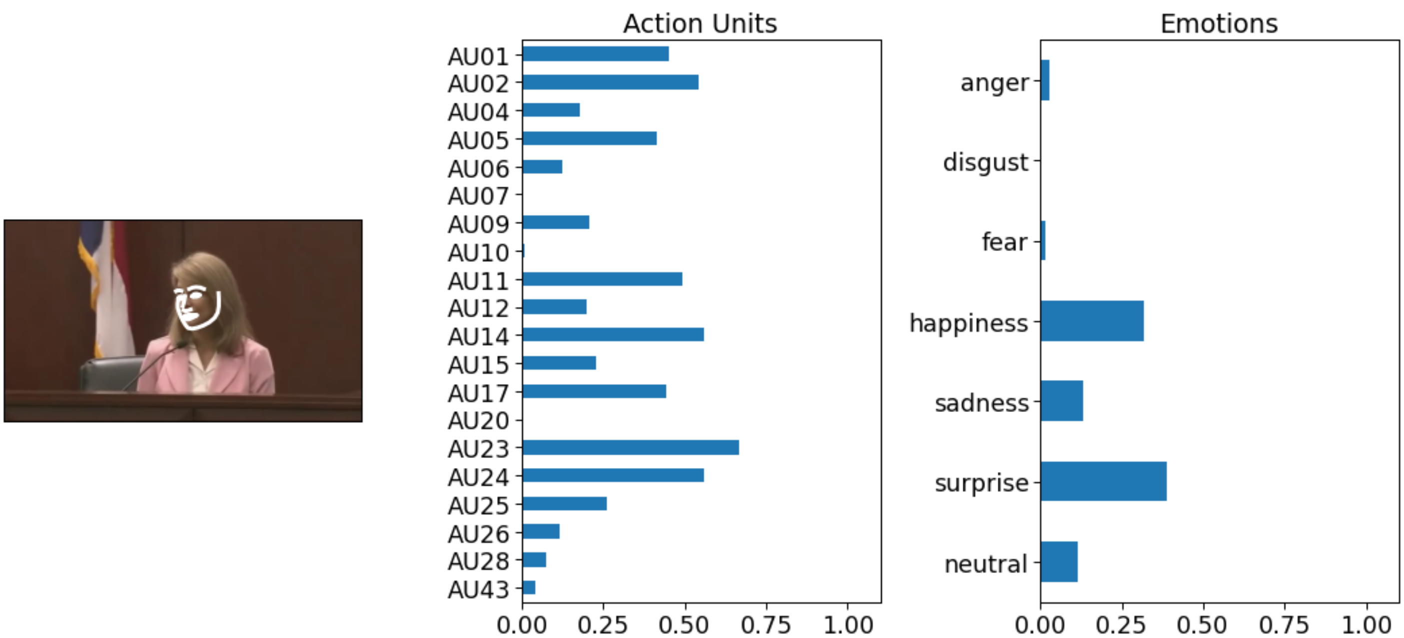
frame	FaceRectX	FaceRectY	FaceRectWidth	FaceRectHeight	FaceScore	x_0	x_1	x_2	
0	407.422024	112.832275	98.081231	125.438125	0.989365	412.103813	411.727582	413.331898	417.7
24	406.772099	117.377222	94.385830	133.057894	0.988867	414.756456	413.050454	413.048499	415.8
48	406.853435	116.365819	93.801358	134.381522	0.983064	415.042079	413.527325	413.565464	416.0
72	407.849892	119.791966	96.051776	132.960582	0.987813	417.055838	415.178728	415.232479	418.2
96	406.160106	124.914687	95.842053	129.252791	0.986462	415.824453	414.193605	414.493357	417.7

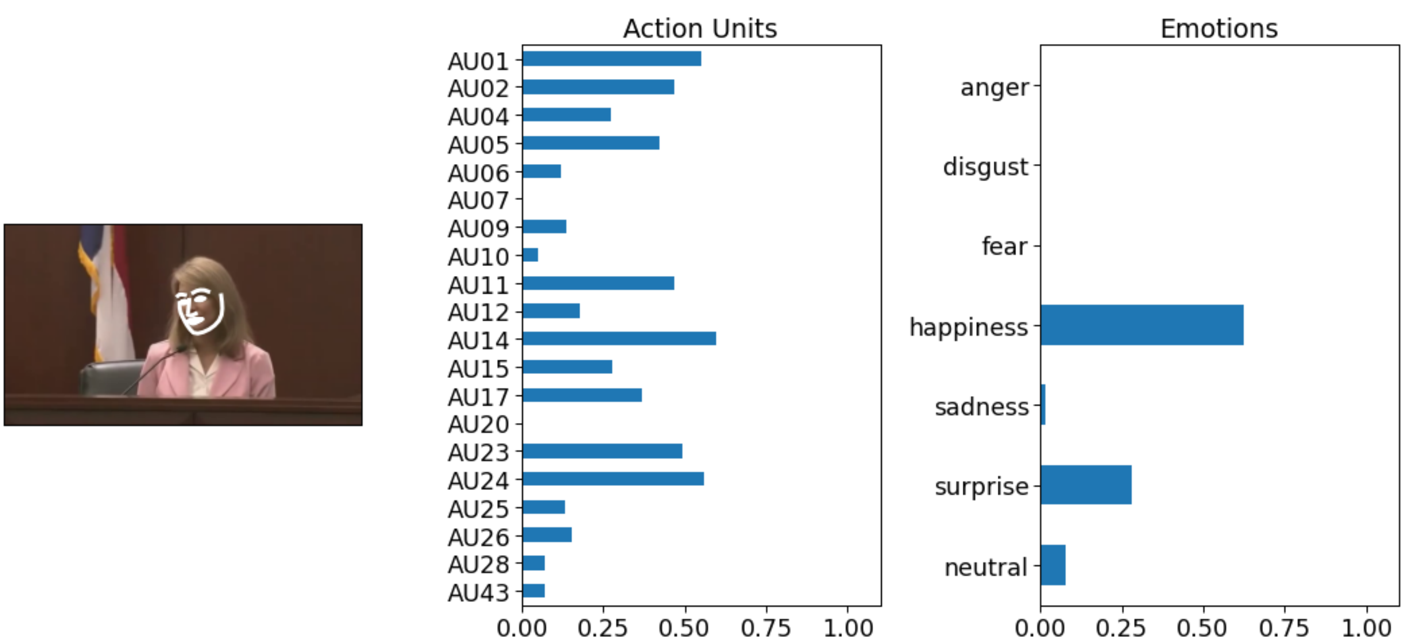
5 rows × 173 columns

In [15]:
```
#plot the graphs for one video
video_prediction.plot_detections(faceboxes=False, add_titles=False)
```

Out[15]:
```
[<Figure size 1080x504 with 3 Axes>,
 <Figure size 1080x504 with 3 Axes>,
 <Figure size 1080x504 with 3 Axes>,
 <Figure size 1080x504 with 3 Axes>,
 <Figure size 1080x504 with 3 Axes>,
 <Figure size 1080x504 with 3 Axes>]
```

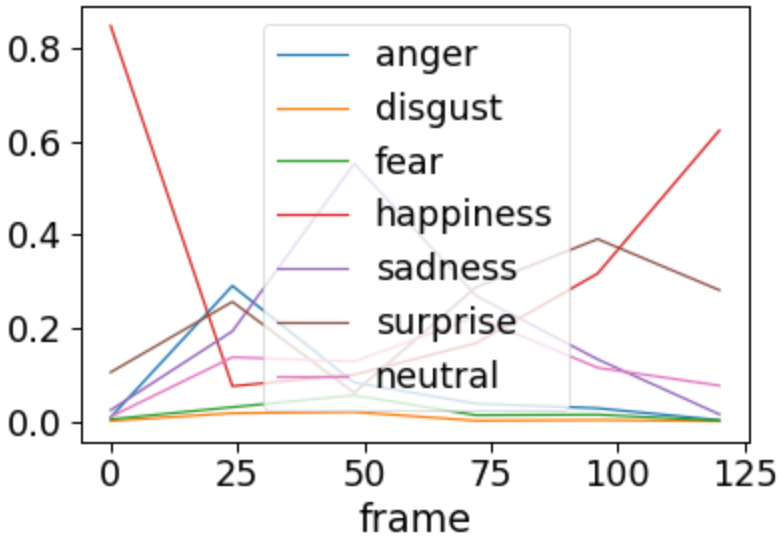






In [16]:

```
#plot the emotion graph
axes = video_prediction.emotions.plot()
```



In [6]:

```
## generate the output for all the videos
from glob import glob
import pandas as pd
import numpy as np
import os

videos = np.sort(glob("C:\\Work\\606Capstone\\Video_chunks\\Sample3\\*.mp4"))
for video in tqdm(videos):
    out_name = video.replace(".mp4", ".csv")
    if not os.path.exists(out_name):

        print(f"Processing: {video}")

        # This is the line that does detection!
        fex = detector.detect_video(video)

        fex.to_csv(out_name, index=False)
```

```
  0%|
      | 0/5 [00:00<?, ?it/s]
Processing: C:\Work\606Capstone\Video_chunks\Sample3\trial_truth_060_000.mp4
  0%|
    | 0/108 [00:00<?, ?it/s]
  1%|▉                                                                                |
1/108 [00:05<09:10,  5.15s/it]
  2%|█▊                                                                               |
2/108 [00:10<09:08,  5.18s/it]
  3%|██▋                                                                              |
3/108 [00:15<09:04,  5.18s/it]
  4%|███▌                                                                             |
4/108 [00:20<08:58,  5.18s/it]
  5%|████▍                                                                            |
5/108 [00:25<08:47,  5.12s/it]
  6%|█████▎                                                                           |
6/108 [00:30<08:43,  5.13s/it]
  6%|██████▏                                                                          |
7/108 [00:36<08:42,  5.17s/it]
  7%|███████                                                                          |
8/108 [00:41<08:30,  5.11s/it]
  8%|████████                                                                         |
9/108 [00:46<08:24,  5.10s/it]
  9%|████████▉                                                                        | 1
0/108 [00:51<08:19,  5.10s/it]
 10%|█████████▊                                                                       | 1
1/108 [00:56<08:18,  5.14s/it]
```

```
 89%|                                                                    | 9
6/108 [08:16<01:02,  5.21s/it]
 90%|                                                                    | 9
7/108 [08:21<00:57,  5.20s/it]
 91%|                                                                    | 9
8/108 [08:26<00:51,  5.16s/it]
 92%|                                                                    | 9
9/108 [08:31<00:45,  5.09s/it]
 93%|                                                                   | 10
0/108 [08:36<00:40,  5.06s/it]
 94%|                                                                   | 10
1/108 [08:41<00:35,  5.07s/it]
 94%|                                                                   | 10
2/108 [08:47<00:30,  5.12s/it]
 95%|                                                                   | 10
3/108 [08:52<00:25,  5.10s/it]
 96%|                                                                   | 10
4/108 [08:57<00:20,  5.11s/it]
 97%|                                                                   | 10
5/108 [09:02<00:15,  5.13s/it]
 98%|                                                                   | 10
6/108 [09:07<00:10,  5.10s/it]
 99%|                                                                   | 10
7/108 [09:12<00:05,  5.07s/it]
100%|                                                                  | 10
8/108 [09:17<00:00,  5.16s/it]
100%|                                                                  |
5/5 [46:55<00:00, 563.16s/it]
```

In [9]:

```
#Merge all the excels for having one file as a feature file
from glob import glob
import pandas as pd
import numpy as np
import os

excels = np.sort(glob("C:\\Work\\606Capstone\\Video_chunks\\Excel\\*.csv"))

#Aggregate detections using a Fex dataframe
from feat.utils.io import read_feat

fex = pd.concat(map(lambda excel: read_feat(excel), excels))

print(f"Unique videos: {fex.inputs.nunique()}")
print(f"Total processed frames: {fex.shape[0]}")
print(f"Avg frames per video: {fex.groupby('input').size().mean()}")
```

```
Unique videos: 742
Total processed frames: 98486
Avg frames per video: 132.73045822102426
```

In [11]:

```
fex.head()
```

Out[11]:

	FaceRectX	FaceRectY	FaceRectWidth	FaceRectHeight	FaceScore	x_0	x_1	x_2	x_3
0	407.422024	112.832275	98.081231	125.438125	0.989365	412.103813	411.727582	413.331898	417.74058
1	407.379583	112.840490	98.065050	125.461160	0.989362	412.170298	411.798642	413.402133	417.79853
2	407.730491	113.668805	98.572132	125.491949	0.987956	412.787646	412.444740	414.060805	418.35748
3	407.656019	113.863420	98.482440	125.406488	0.987745	413.093359	412.650282	414.318238	418.86756
4	408.283298	114.822023	98.516228	124.739104	0.986979	413.305362	412.989890	414.624331	418.99639

5 rows × 173 columns

In [12]:

```
#summarize the data (get mean)
by_video = fex.update_sessions(fex["input"])

# Compute the mean per video
video_means = by_video.extract_mean()

video_means[['mean_AU01','mean_AU02','mean_AU04','mean_AU05','mean_AU06','mean_AU07','mean
```

Out[12]:

	mean_AU01	mean_AU02	mean_AU04	mean_AU0
C:\Work\606Capstone\Video_chunks\Sample2\trial_lie_007_000.mp4	0.331584	0.412600	0.343551	0.56004
C:\Work\606Capstone\Video_chunks\Sample2\trial_lie_007_001.mp4	0.382833	0.433317	0.392617	0.52414
C:\Work\606Capstone\Video_chunks\Sample2\trial_lie_007_002.mp4	0.357702	0.410314	0.340423	0.51281
C:\Work\606Capstone\Video_chunks\Sample2\trial_lie_007_003.mp4	0.332924	0.430573	0.316183	0.42828
C:\Work\606Capstone\Video_chunks\Sample2\trial_lie_007_004.mp4	0.323526	0.455067	0.397633	0.44062
...	...	...	...	
C:\Work\606Capstone\Video_chunks\Sample\trial_truth_059_002.mp4	0.367617	0.475653	0.397558	0.36057
C:\Work\606Capstone\Video_chunks\Sample\trial_truth_059_003.mp4	0.432222	0.456469	0.389874	0.33762
C:\Work\606Capstone\Video_chunks\Sample\trial_truth_059_004.mp4	0.342524	0.433497	0.433546	0.32385
C:\Work\606Capstone\Video_chunks\Sample\trial_truth_059_005.mp4	0.346360	0.444096	0.416365	0.34021
C:\Work\606Capstone\Video_chunks\Sample\trial_truth_059_006.mp4	0.352154	0.454667	0.405486	0.35057

742 rows × 27 columns