

```
In [26]: %load_ext do_not_print_href
```

The `do_not_print_href` extension is already loaded. To reload it, use:

```
%reload_ext do_not_print_href
```

## Motion Generation Example using RNN

- “Generative Choreography using Deep Learning”, Luka Crnkovic-Friis, Louise Crnkovic-Friis, 2016 (arXiv:1605.06921 (<https://arxiv.org/abs/1605.06921>))

This article will be presented at the 7<sup>th</sup> International Conference on Computational Creativity, ICC2016

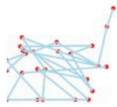
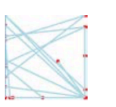
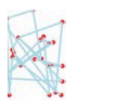






Training Time	Sample frames from generated animation			Description
~10 min				Nearly untrained system. Joint positions are almost random. <a href="https://www.youtube.com/watch?v=QnaKyc1Mpmo">https://www.youtube.com/watch?v=QnaKyc1Mpmo</a>
~6h				Understands relative joint positions and very basic movement. <a href="https://www.youtube.com/watch?v=c9h9zc7uPWQ">https://www.youtube.com/watch?v=c9h9zc7uPWQ</a>
~48h				Understands joint relations well, understand syntax and style well, understands basic semantics <a href="https://www.youtube.com/watch?v=Q4_XSMqN8w0">https://www.youtube.com/watch?v=Q4_XSMqN8w0</a> <a href="https://www.youtube.com/watch?v=W1oRgDPxEkc">https://www.youtube.com/watch?v=W1oRgDPxEkc</a>

Table 1 Example results over time

# motion capture data sample

- CMU motion caption data
  - <http://mocap.cs.cmu.edu> (<http://mocap.cs.cmu.edu>)
  - "Subject #94 (indian dance)"

# CMU Graphics Lab Motion Capture Database

[Home](#)
[Search](#)
[Tools](#)
[Info](#)
[FAQs](#)
[Rendered Movies](#)
[Resources](#)

View All:

[Subjects](#)
[Motions](#)

Browse:

[Motion Categories](#)

Search Help

subject number

motion or keyword

(e.g. 41)

(e.g. run)

Subject #94 (indian dance) <a href="#">file index</a>			-- asf			framerate	Feedback
Image	Trial #	Motion Description					
	1		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	2		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	3		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	4		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	5		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	6		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	7		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	8		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	9		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	10		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	11		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	12		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	13		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	14		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	15		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>
	16		<a href="#">c3d</a>	<a href="#">amc</a>	<a href="#">Animated</a>	120	<a href="#">Feedback</a>

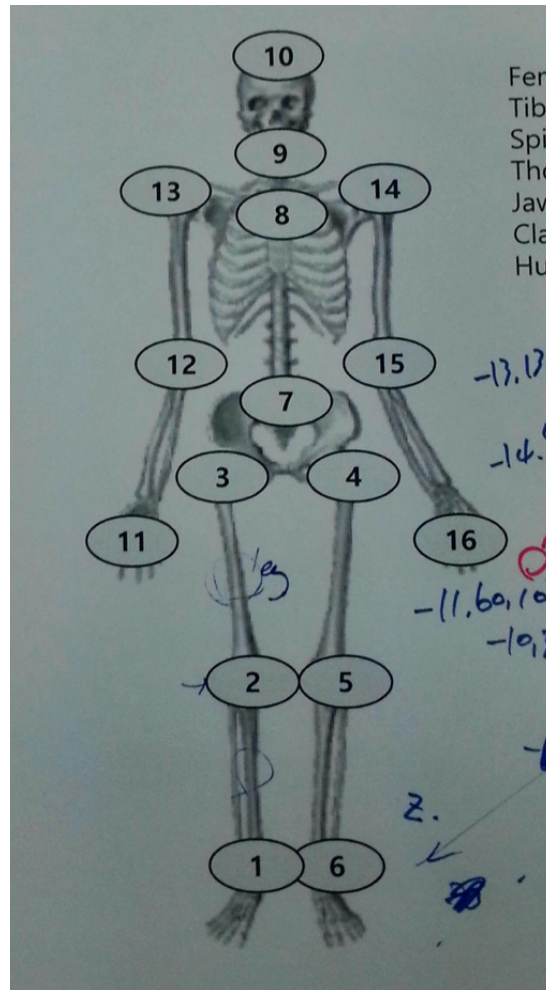
```
In [27]: %%bash
rm -fr data/input
rm -fr data/input2 data/input3
rm -fr data/input4 data/tmp
rm -fr save1 save2 save3 save4
```

```
In [28]: %matplotlib inline
# coding: utf-8
from __future__ import print_function
from __future__ import division
from __future__ import absolute_import
import numpy as np
import matplotlib.pyplot as plt
```

## 모션데이터를 전처리 하여 csv 포맷으로 변환

- 모션 캡처 데이터의 한 프레임에서 주요 관절의 3차원 좌표값 추출하여 차례로 기록
- 모션 캡처 데이터 모든 프레임에 대해서 위 과정을 반복
- 주요관절의 아래 그림과 같이 선택
- csv 데이터의 하나의 row 에는 16개 관절 좌표의 x,y,z 값을 차례대로 기록 (전체 48개 column)

$[x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_{16}, y_{16}, z_{16}]$



## 모션 데이터 예시

- data/mocap-thkim-3d/94\_04\_skeleton\_3d.csv

```
In [29]: data = np.loadtxt(  
          'data/mocap-thkim-3d/94_04_skeleton_3d.csv',  
          delimiter=',')  
data.shape
```

Out[29]: (716, 48)

```
In [30]: import pandas as pd
df = pd.DataFrame(data)
colnames = [a+str(b)
              for b in range(1,17)
              for a in ['x_', 'y_', 'z_'] ]
df.columns = colnames
df
```

Out[30]:

	x_1	y_1	z_1	x_2	y_2	z_2	x_3
0	1.882192	756.66960	144.147340	-31.925642	697.06940	107.631160	2.9472
1	1.883342	756.67890	144.150670	-31.815847	697.03030	107.629860	2.9253
2	1.890642	756.70060	144.164610	-31.691278	696.98720	107.639490	2.8684
3	1.888782	756.71560	144.184250	-31.598267	696.93536	107.655106	2.7869
4	1.897167	756.73083	144.202870	-31.539127	696.90410	107.680260	2.6528
5	1.907214	756.74603	144.215380	-31.534208	696.90234	107.715510	2.4287
6	1.917874	756.75354	144.212880	-31.573100	696.93710	107.697930	2.1183
7	1.939046	756.76056	144.198700	-31.622274	697.04020	107.598000	1.6795
8	1.955456	756.77966	144.174730	-31.690561	697.20610	107.395930	1.0666
9	1.951021	756.83026	144.160140	-31.695585	697.41925	107.128660	0.3110
10	1.947459	756.90656	144.148560	-31.602858	697.62616	106.830560	-0.538
11	1.938678	756.97420	144.121980	-31.450668	697.77850	106.487724	-1.406
12	1.900310	757.03460	144.102630	-31.214990	697.88257	106.088980	-2.278
13	1.831522	757.10370	144.107040	-30.911634	697.96640	105.642820	-3.108
14	1.747009	757.16330	144.123520	-30.636566	698.02970	105.209900	-3.822
15	1.678542	757.20940	144.133700	-30.414112	698.08966	104.865600	-4.276
16	1.624095	757.26544	144.152500	-30.169820	698.15314	104.586464	-4.361
17	1.584209	757.31714	144.174930	-29.792175	698.18365	104.234184	-4.144
18	1.563949	757.34450	144.175740	-29.256231	698.18866	103.760130	-3.742
19	1.561036	757.33887	144.151430	-28.647322	698.15420	103.267140	-3.126
20	1.578463	757.29010	144.114550	-27.984106	698.06024	102.815880	-2.117
21	1.603456	757.17633	144.057170	-27.196768	697.86960	102.292830	-0.718
22	1.711625	757.04980	143.954640	-26.159270	697.53064	101.820490	0.8030
23	1.830794	756.87150	143.934130	-24.984285	696.89820	101.690500	2.4037
24	1.723435	756.22520	143.869340	-23.844180	695.75610	101.555810	4.2539
25	1.499542	754.62980	143.472550	-22.809896	694.18330	100.606636	6.2052
26	1.337029	752.14417	142.335740	-22.206560	692.32690	98.185200	7.9170

27	1.265661	749.50340	140.172490	-21.823687	690.36945	94.723820	9.1957
28	1.366566	747.38696	136.877320	-21.139654	688.50165	90.791540	10.078
29	1.719596	746.15936	132.191200	-19.965536	686.71860	86.430275	10.606
...	...	...	...	...	...	...	...
686	-2.531506	753.10020	-57.438564	-26.558670	687.00610	-89.963440	-0.447
687	-3.804404	750.34160	-58.423798	-29.106009	685.63214	-93.178790	-0.815
688	-6.878615	744.68100	-61.351900	-34.576366	682.40560	-99.473340	-0.755
689	-12.088651	737.63190	-66.647080	-41.752422	677.45325	-107.078896	-0.516
690	-19.129330	732.52590	-74.027054	-48.510345	671.79090	-113.641106	-0.548
691	-28.785208	732.38710	-84.101470	-54.068897	666.73760	-117.944820	-1.176
692	-41.993830	737.06350	-97.422080	-57.927338	663.93560	-119.946670	-2.050
693	-57.686380	743.29016	-113.234940	-59.667873	665.05396	-120.329810	-3.482
694	-72.226555	747.91010	-128.932540	-59.542133	670.47420	-119.649230	-5.983
695	-80.439670	750.79320	-139.513460	-58.328766	677.88556	-118.601890	-9.668
696	-80.685425	754.60300	-142.215590	-56.419647	684.10376	-117.996150	-14.41
697	-80.060790	756.99896	-142.093700	-55.020363	687.61420	-116.862686	-19.43
698	-81.515100	757.58210	-141.709270	-56.411854	688.08276	-116.190020	-23.23
699	-82.678360	757.92520	-141.409030	-59.363533	687.25195	-116.942080	-25.08
700	-83.172844	757.79016	-141.864290	-61.196210	686.55426	-117.820310	-25.64
701	-83.001910	757.49840	-142.166170	-62.252880	685.72840	-118.972560	-25.70
702	-82.220085	757.17487	-142.034130	-64.465385	684.07180	-120.494100	-25.34
703	-80.345310	756.49603	-141.142720	-68.430830	681.03510	-122.637690	-24.74
704	-76.388920	755.01373	-137.853800	-72.429860	677.17554	-125.607925	-24.47
705	-68.397280	752.79550	-130.502910	-74.868576	674.10660	-128.878020	-24.90
706	-57.514492	749.99180	-120.290530	-74.909386	673.48010	-131.119840	-26.08
707	-47.540966	747.37620	-109.833390	-72.263390	675.66470	-132.186690	-27.63
708	-40.426094	747.27680	-101.572260	-67.227165	679.96210	-132.474000	-29.29
709	-36.221110	751.35950	-97.827810	-60.754690	684.89710	-131.145570	-31.15
710	-33.827175	755.21124	-96.917410	-55.527416	687.78740	-129.189500	-32.89
711	-33.839127	755.76086	-97.059310	-53.661446	687.41223	-129.745060	-34.06
712	-34.591507	755.82430	-97.278010	-53.734493	686.72626	-130.451550	-34.88
713	-34.687830	756.16090	-97.028250	-54.017150	687.08520	-129.065670	-36.09
714	-34.806538	756.39764	-96.819695	-54.563060	687.28910	-127.727580	-37.77

715	-34.987312	756.55720	-96.684940	-55.357400	687.19183	-127.087610	-39.34
-----	------------	-----------	------------	------------	-----------	-------------	--------

716 rows × 48 columns



In [31]: `df.describe()`

Out[31]:

	x_1	y_1	z_1	x_2	y_2	z_2
count	716.000000	716.000000	716.000000	716.000000	716.000000	716.000000
mean	-3.518349	741.012391	60.978439	-19.462976	679.968872	32.958518
std	165.634351	26.224731	89.881845	156.700390	22.239540	82.227495
min	-247.948400	597.344400	-142.215590	-250.783620	573.122800	-132.474000
25%	-112.370794	738.313720	-4.724420	-122.785225	675.907575	-31.781075
50%	-2.874052	751.471200	81.822103	-27.833915	685.672270	44.818014
75%	10.579322	756.158225	118.554532	-8.219873	692.030925	100.865803
max	436.693760	759.918150	214.190540	420.481050	711.845030	176.197020

8 rows × 48 columns



## 모션데이터 시각화 - 주요 좌표들의 변화 plot

- 신체중심 (골반) x,y,z 좌표의 변화
- 기타 주요 단말 (손,발,정수리) x,y,z 좌표의 변화
- 200 스텝까지의 변화와, 그 이후 데이터 전체의 변화를 관찰

```
In [32]: def plot_motion(data, vlim=5.5):
    if type(data) is str or type(data) is unicode:
        data = np.loadtxt(data, delimiter=',')

    cols0 = [18,19,20] # 골반 = (7)

    cols = np.concatenate([
        [0,1,2],      # 오른쪽 발 = (0)
        [15,16,17],   # 왼쪽 발 = (6)
        [27,28,29],   # 정수리 = (10)
        [30,31,32],   # 오른손 = (11)
        [45,46,47],   # 왼손 = (16)
    ])

    plt.figure(figsize=(13.5, 3.5))

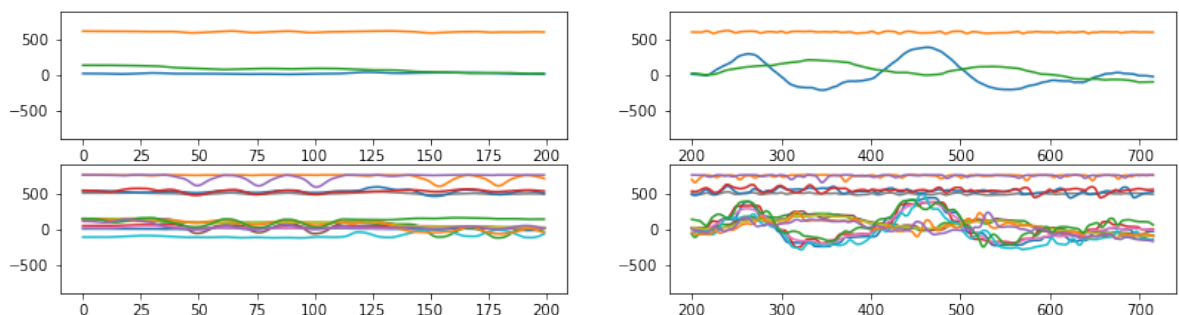
    # 앞에서부터 500 스텝만 plot
    plt.subplot(2,2,1)
    plt.ylim(-vlim, vlim)
    plt.plot(range(200), data[:200, cols0])

    plt.subplot(2,2,3)
    plt.ylim(-vlim, vlim)
    plt.plot(range(200), data[:200, cols])

    # 나머지 스텝 plot
    plt.subplot(2,2,2)
    plt.ylim(-vlim, vlim)
    plt.plot(range(200, len(data)), data[200:, cols0])

    plt.subplot(2,2,4)
    plt.ylim(-vlim, vlim)
    plt.plot(range(200, len(data)), data[200:, cols])
```

```
In [33]: plot_motion('data/mocap-thkim-3d/94_04_skeleton_3d.csv',
    vlim=900)
```

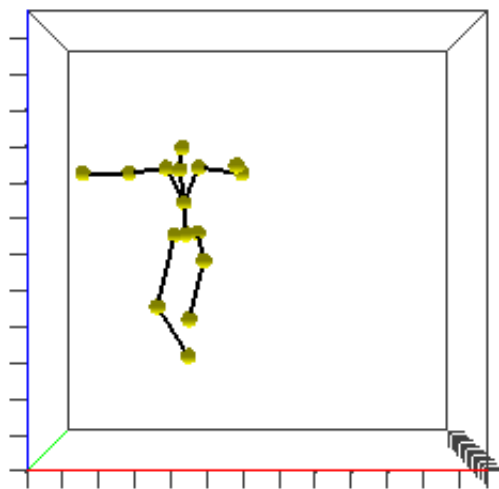


## 모션 데이터 시각화 - 애니메이션

- Javascript + Processing.js 기반의 간단한 3D 애니메이션 뷰어
- Jupyter Notebook 의 HTML 렌더링 기능을 이용

```
In [34]: def show_motion(filename):  
        from IPython.core.display import HTML, display  
        html="""  
<iframe  
        style="width:440px;height:440px;border:0"  
        src="BodyViewerJs/index.html#../{filename:s}">  
</iframe>  
        """  
        display(HTML(html.format(filename=filename)))
```

```
In [35]: show_motion('data/mocap-thkim-3d/94_04_skeleton_3d.csv')
```



```
In [36]: !ls data/mocap-thkim-3d/
```

```
94_01_skeleton_3d.csv  94_05_skeleton_3d.csv  94_09_skeleton_3d.cs  
v  
94_02_skeleton_3d.csv  94_06_skeleton_3d.csv  94_10_skeleton_3d.cs  
v  
94_03_skeleton_3d.csv  94_07_skeleton_3d.csv  94_11_skeleton_3d.cs  
v  
94_04_skeleton_3d.csv  94_08_skeleton_3d.csv  94_12_skeleton_3d.cs  
v
```



```
In [37]: %%html
<a href="catalog.html" target="_">
click me!
</a>
```

[click me! \(catalog.html\)](#)

## 학습을 위해서 학습 데이터를 정규화 하고, data/input 에 저장

- step #1: get means & standard deviations for (multiple) input files
  - 입력파일(들) 에 포함된 좌표값들의 평균, 표준편차 계산 (관절별로 따로)
- step #2:
  - 위에서 구한 평균 분산을 이용해서 입력파일의 좌표를 변환
  - (좌표값 - 평균값) / 표준편차

```
In [38]: # %pycat csv_stats.py
```

```
In [39]: %%bash -e
# step #1: get means & standard deviations
python csv_stats.py \
    data/mocap-thkim-3d/94_04_skeleton_3d.csv \
    --mean_file data/input/mean.txt \
    --std_file data/input/std.txt
```

```
{'var_file': None, 'verbose': False, 'std_file': 'data/input/std.t
xt', 'mean_file': 'data/input/mean.txt', 'input_files': ['data/moc
ap-thkim-3d/94_04_skeleton_3d.csv']}
input: data/mocap-thkim-3d/94_04_skeleton_3d.csv
wrote: data/input/mean.txt
wrote: data/input/std.txt
```

```
In [40]: # %pycat csv_normalize.py
```

```
In [41]: %%bash -e
# step #2: normalize
# (1) subtract with mean value
# (2) divide by std value
python csv_normalize.py \
    --mean_file data/input/mean.txt \
    --std_file data/input/std.txt \
    data/mocap-thkim-3d/94_04_skeleton_3d.csv \
    data/input/04.csv
```

```
{'scale': 1.0, 'std_file': 'data/input/std.txt', 'reverse': False,
'input_file': 'data/mocap-thkim-3d/94_04_skeleton_3d.csv', 'output
_file': 'data/input/04.csv', 'mean_file': 'data/input/mean.txt', '
verbose': False}
normalize: data/mocap-thkim-3d/94_04_skeleton_3d.csv data/input/04
.csv
```

## 역정규화 un-normalize

- normalize step
  - $(\text{좌표값} - \text{평균값}) / \text{표준편차}$
- un-normalize step (reverse normalize)
  - $\text{변환좌표값} * \text{표준편차} + \text{평균값}$

```
In [42]: %%bash -e
# un-normalize step (reverse normalize)
# (1) multiply by std value
# (2) add mean value
python csv_normalize.py -r \
    --mean_file data/input/mean.txt \
    --std_file data/input/std.txt \
    data/input/04.csv \
    data/tmp/rev-04.csv

{'scale': 1.0, 'std_file': 'data/input/std.txt', 'reverse': True,
 'input_file': 'data/input/04.csv', 'output_file': 'data/tmp/rev-04.csv', 'mean_file': 'data/input/mean.txt', 'verbose': False}
un-normalize: data/input/04.csv data/tmp/rev-04.csv
```

## 변환 + 역변환 결과가 원본과 같은지 검사

```
In [43]: %%bash -e
# compare original data with un-normalized data
python csv_compare.py \
    data/mocap-thkim-3d/94_04_skeleton_3d.csv \
    data/tmp/rev-04.csv

{'input2': 'data/tmp/rev-04.csv', 'input1': 'data/mocap-thkim-3d/94_04_skeleton_3d.csv'}
max square error: 6.72400000334e-09
```

## 훈련데이터의 구성

```
In [44]: # %pycat dataloader.py
```

## prepare training data - DataLoader (dataloader.py)

```
for dirname, _, filelist in os.walk(data_dir):
    for filename in filelist:
        filepath = join(dirname, filename)
        if filename.endswith('.csv'):
            logger.info(('loadtxt', filepath))

            data = np.loadtxt(filepath, delimiter=',')

            if augment_data > 1:
                # mirror augment * xN augment
                data = np.vstack([data, data[::-1]] * augment_data)
                logger.info(('augment > 1', 'shape', data.shape))

            self.data.append(data)

self.data_len = [len(d) for d in self.data]
self.data_prob = np.array(self.data_len, dtype=np.float32) / \
    np.sum(self.data_len, dtype=np.float32)
```

## prepare training data batch - DataLoader (dataloader.py)

```
for _ in range(batch_size):
    data_choice = np.random.choice(range(len(self.data)), p=self.data_prob)

    data = self.data[data_choice]
    index = np.random.randint(0, len(data) - (seq_length + 1))
    x = data[index:index + seq_length, :]
    y = data[index + 1:index + 1 + seq_length, :]
    x_batch.append(x)
    y_batch.append(y)

    data_id = self.data_id[data_choice]
    id_batch.append(data_id)
```

## 생성모형 정의

```
In [45]: # %pycat model.py
```

## placeholders

```
input_data = tf.placeholder(
    dtype=tf.float32,
    shape=[None, seq_length, NUM_OUTPUTS],
    name='input_data')
target_data = tf.placeholder(
    dtype=tf.float32,
    shape=[None, seq_length, NUM_OUTPUTS],
    name='target_data')
seq_length = tf.placeholder(
    dtype=tf.int64,
    shape=[None],
    name='seq_length')
motion_id = tf.placeholder(
    dtype=tf.float32,
    shape=[None, ID_SIZE],
    name='mot_id')
batch_size = tf.shape(seq_length)[0]
```

## RNN

```
cell_list = \
    [new_cell(args, NUM_OUTPUTS, infer)] + \
    [new_cell(args, rnn_size, infer)] * (num_layers-1)
cell = tf.contrib.rnn.MultiRNNCell(cell_list)

motion_state = tf.layers.dense(motion_id,
                                rnn_size * 2 * num_layers)
initial_state = tuple([
    tf.contrib.rnn.LSTMStateTuple(*tf.split(x, 2, axis=1))
    for x in tf.split(motion_state, num_layers, axis=1)])

outputs, last_state = tf.nn.dynamic_rnn(
    cell,
    input_data,
    sequence_length=seq_length,
    initial_state = initial_state,
    dtype=tf.float32
)
output = tf.layers.dense(outputs, NUM_OUTPUTS)
```

## Loss & Optimize

```
# loss A MSE on relative coords
loss = tf.losses.mean_squared_error(
    target_data,
    output)
```

## Sampling

```
def sample(model, sess, num):
```

```
    prev_state    = sess.run(model.initial_state, {model.motion_id: [motion_id]})
    prev_vec      = np.zeros((1, 1, NUM_OUTPUTS), dtype=np.float32)
    strokes       = np.zeros((num, NUM_OUTPUTS), dtype=np.float32)
    output        = np.zeros(NUM_OUTPUTS, dtype=np.float32)

    for i in tqdm(range(num)):
        feed      = {
            model.input_data:    prev_vec,
            model.initial_state: prev_state,
            model.seq_length:    [1],
        }
        o_rest, next_state = sess.run(
            [model.output, model.last_state],
            feed)
        strokes[i, :]      = o_rest[0]
        prev_vec[0, 0, :]  = o_rest[0, 0, :]
        prev_state         = next_state

    return strokes
```

</code>

## 학습 시작

In [46]:

```
# "Last executed 2017-10-10 22:06:10 in 10m 40s"
!python -u train.py \
  --rnn_type lstm \
  --data_dir data/input \
  --save_dir save1
```

```
INFO:dataloader:('checking:', 'data/input')
INFO:dataloader:('loadtxt', 'data/input/04.csv')
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 10
0)
epoch 1, step 200, loss = 0.20350, elapsed = 0.108
epoch 1, step 400, loss = 0.08182, elapsed = 0.114
model saved to save1/model.ckpt-500
epoch 1, step 600, loss = 0.03780, elapsed = 0.120
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 10
0)
epoch 2, step 800, loss = 0.02288, elapsed = 0.110
epoch 2, step 1000, loss = 0.01247, elapsed = 0.130
model saved to save1/model.ckpt-1000
epoch 2, step 1200, loss = 0.00892, elapsed = 0.243
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 10
0)
epoch 3, step 1400, loss = 0.00727, elapsed = 0.113
model saved to save1/model.ckpt-1500
epoch 3, step 1600, loss = 0.00580, elapsed = 0.113
epoch 3, step 1800, loss = 0.00447, elapsed = 0.120
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 10
0)
epoch 4, step 2000, loss = 0.00357, elapsed = 0.107
model saved to save1/model.ckpt-2000
epoch 4, step 2200, loss = 0.00355, elapsed = 0.119
epoch 4, step 2400, loss = 0.00701, elapsed = 0.114
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 10
0)
model saved to save1/model.ckpt-2500
epoch 5, step 2600, loss = 0.00235, elapsed = 0.109
epoch 5, step 2800, loss = 0.00256, elapsed = 0.109
epoch 5, step 3000, loss = 0.00187, elapsed = 0.125
model saved to save1/model.ckpt-3000
model saved to save1/model.ckpt-3080
```

In [47]:

```
#!/python -mtensorboard.main --port 5000 --logdir save1:save1,save2:save2,
save3:save3
```

## 학습된 네트워크를 이용해서 동작 생성

In [48]:

```
!python sample.py \
--save_dir save1 \
--motion_id 0,0,0,0,1 \
--output_file data/tmp/sample_04.csv
```

```
loading model: save1/model.ckpt-3080  
100%|██████████████████████████████████████| 1000/1000 [00:01<00:00, 566.3  
7it/s]
```

정규화된 입력을 학습했기 때문에, 사용하기 전에는 다시 역-정규화



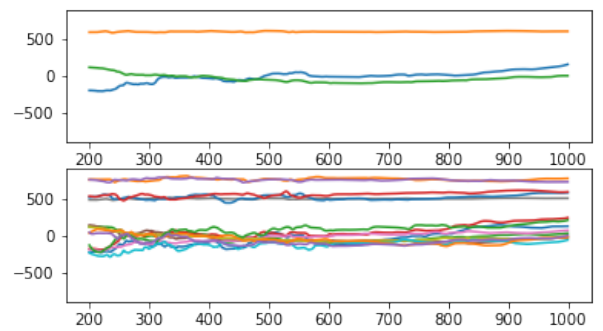
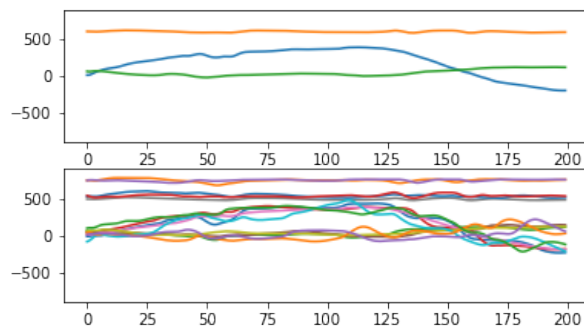
In [49]:

```
!python csv_normalize.py -r \N
  --mean_file data/input/mean.txt \
  --std_file data/input/std.txt \
  data/tmp/sample_04.csv \
  save1/sample_04.csv
```

```
{'scale': 1.0, 'std_file': 'data/input/std.txt', 'reverse': True, 'input_f
ile': 'data/tmp/sample_04.csv', 'output_file': 'save1/sample_04.csv', 'mea
n_file': 'data/input/mean.txt', 'verbose': False}
un-normalize: data/tmp/sample_04.csv save1/sample_04.csv
```

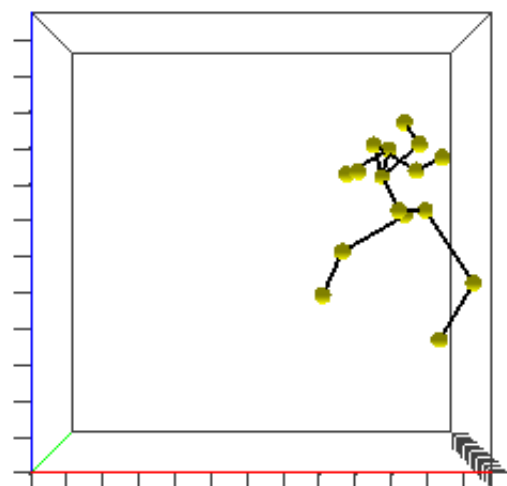
In [50]:

```
plot_motion('save1/sample_04.csv',  
            vlim=900.0)
```



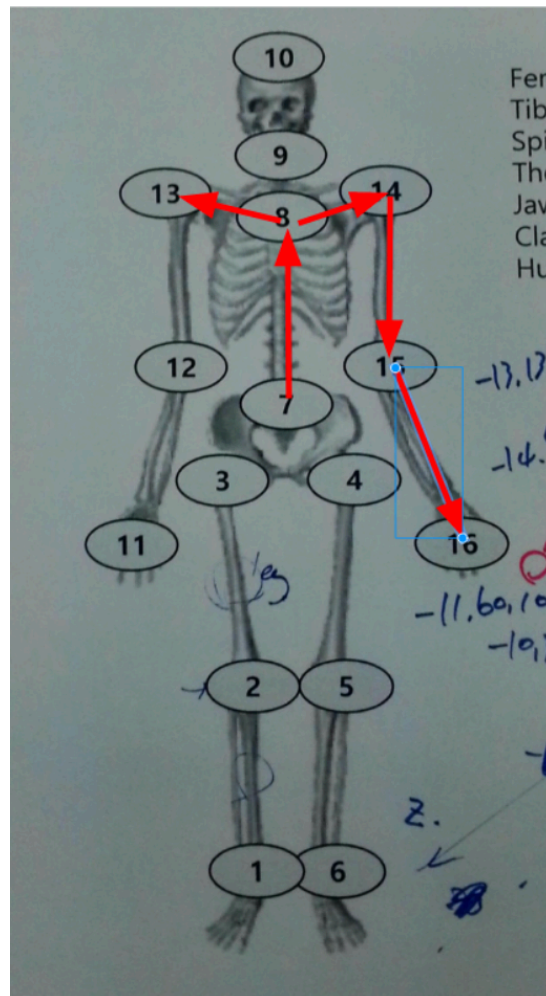
In [51]:

```
show_motion('save1/sample_04.csv')
```



# 데이터 포맷의 변경

- 인체 중심을 골반 (7번) 관절 기준으로
- 연결된 두 개 관절에서 중심 방향에 있는 관절을 기준으로
- 바깥 방향에 있는 관절은 중심방향에 있는 관절에 대한 상대좌표
- 중심좌표는 절대값을 그대로 사용



In [52]:

```
# %pycat csv_motion_to_jrel2.py
```

```

LINKS = (
    (7, 3), (3, 2), (2, 1), # 골반에서 우측 발까지
    (7, 4), (4, 5), (5, 6), # 골반에서 좌측 발까지
    (7, 8), (8, 9), (9, 10), # 골반에서 정수리까지
    (8, 13), (13, 12), (12, 11), # 명치에서 우측 손 끝 까지
    (8, 14), (14, 15), (15, 16), # 명치에서 좌측 손 끝 까지
)

```

```

def motion_to_jrel2_row(row):
    row = row[:3*N_JOINTS].reshape([-1,3])
    row_rel = np.zeros_like(row)
    j1, j2 = LINKS[0]
    row_rel[j1-1] = row[j1-1]
    for j1, j2 in LINKS:
        row_rel[j2-1] = row[j2-1] - row[j1-1]
    row = row_rel.reshape([-1])
    return row

```

```

def jrel2_to_motion_row(row):
    row = row[:3*N_JOINTS].reshape([-1,3])
    for j1, j2 in LINKS:
        row[j2-1] = row[j2-1] + row[j1-1]
    row = row.reshape([-1])
    return row

```

</code>

## 상대좌표로 변환하고, 정규화

In [53]:

```
%%bash -e
python csv_motion_to_jrel2.py \
    data/mocap-thkim-3d/94_04_skeleton_3d.csv \
    data/tmp/__rel__.csv

python csv_stats.py \
    --mean_file data/input2/mean.txt \
    --std_file data/input2/std.txt \
    data/tmp/__rel__.csv

python csv_normalize.py \
    --mean_file data/input2/mean.txt \
    --std_file data/input2/std.txt \
    data/tmp/__rel__.csv \
    data/input2/04.csv
```

```
{'output_file': 'data/tmp/__rel__.csv', 'reverse': False, 'input_file': 'data/mocap-thkim-3d/94_04_skeleton_3d.csv'}
{'var_file': None, 'verbose': False, 'std_file': 'data/input2/std.txt', 'mean_file': 'data/input2/mean.txt', 'input_files': ['data/tmp/__rel__.csv']}
input: data/tmp/__rel__.csv
wrote: data/input2/mean.txt
wrote: data/input2/std.txt
{'scale': 1.0, 'std_file': 'data/input2/std.txt', 'reverse': False, 'input_file': 'data/tmp/__rel__.csv', 'output_file': 'data/input2/04.csv', 'mean_file': 'data/input2/mean.txt', 'verbose': False}
normalize: data/tmp/__rel__.csv data/input2/04.csv
```

## 학습시작

In [54]:

```
!python -u train.py \
  --save_dir save2 \
  --rnn_type lstm \
  --data_dir data/input2
```

```
INFO:dataloader:('checking:', 'data/input2')
INFO:dataloader:('loadtxt', 'data/input2/04.csv')
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 1, step 200, loss = 0.32523, elapsed = 0.120
epoch 1, step 400, loss = 0.14858, elapsed = 0.128
model saved to save2/model.ckpt-500
epoch 1, step 600, loss = 0.08465, elapsed = 0.111
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 2, step 800, loss = 0.05817, elapsed = 0.124
epoch 2, step 1000, loss = 0.03374, elapsed = 0.139
model saved to save2/model.ckpt-1000
epoch 2, step 1200, loss = 0.02450, elapsed = 0.132
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 3, step 1400, loss = 0.02208, elapsed = 0.134
model saved to save2/model.ckpt-1500
epoch 3, step 1600, loss = 0.02261, elapsed = 0.125
epoch 3, step 1800, loss = 0.01302, elapsed = 0.132
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 4, step 2000, loss = 0.01438, elapsed = 0.132
model saved to save2/model.ckpt-2000
epoch 4, step 2200, loss = 0.00702, elapsed = 0.138
epoch 4, step 2400, loss = 0.00591, elapsed = 0.101
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
model saved to save2/model.ckpt-2500
epoch 5, step 2600, loss = 0.00580, elapsed = 0.135
epoch 5, step 2800, loss = 0.00467, elapsed = 0.124
epoch 5, step 3000, loss = 0.00405, elapsed = 0.114
model saved to save2/model.ckpt-3000
model saved to save2/model.ckpt-3080
```



```
#!python -mtensorboard.main --port 5000 --logdir save1:save1,save2:save2,save3:save3,save4:save4
```

```
#!python -mtensorboard.main --port 5000 --logdir save1:save1,save2:save2,save3:save3,save4:save4
```

## 학습된 모델을 이용하여 동작 생성

```
!python sample.py \
  --save_dir save2 \
  --motion_id 0,0,0,0,1 \
  --output_file data/tmp/sample_04-2.csv
```

```
!python sample.py \
--save_dir save2 \
--motion_id 0,0,0,0,1 \
--output_file data/tmp/sample_04-2.csv
```

[illegible]

## 역 정규화 (un-normalization)

In [57]:

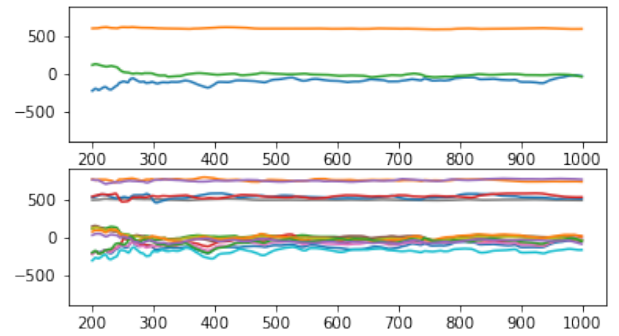
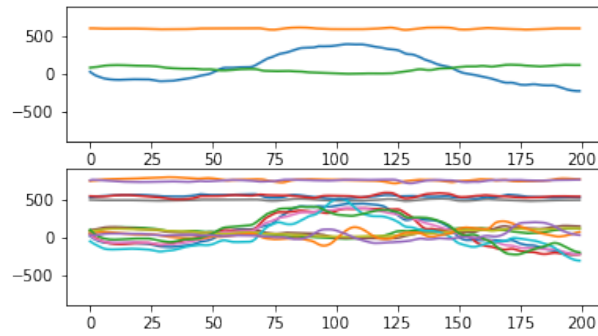
```
%%bash -e
python csv_normalize.py -r \
  --mean_file data/input2/mean.txt \
  --std_file data/input2/std.txt \
  data/tmp/sample_04-2.csv \
  data/tmp/__rel__.csv

python csv_motion_to_jrel2.py -r \
  data/tmp/__rel__.csv \
  save2/sample_04-2.csv
```

```
{'scale': 1.0, 'std_file': 'data/input2/std.txt', 'reverse': True, 'input_file': 'data/tmp/sample_04-2.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input2/mean.txt', 'verbose': False}
un-normalize: data/tmp/sample_04-2.csv data/tmp/__rel__.csv
{'output_file': 'save2/sample_04-2.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
```

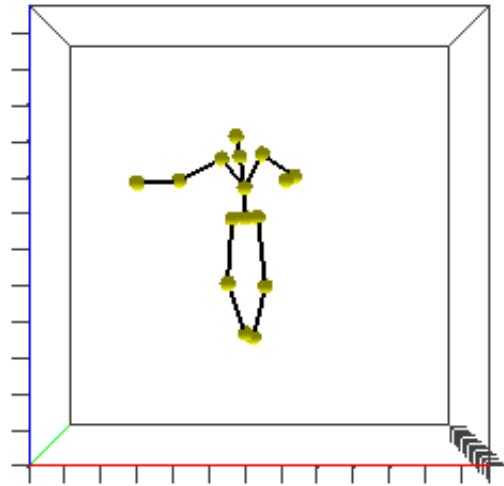
In [58]:

```
plot_motion('save2/sample_04-2.csv',  
            vlim=900.0)
```



In [59]:

```
show_motion('save2/sample_04-2.csv')
```



**Layer-normalized LSTM**

In [60]:

```
!python -u train.py \
  --save_dir save3 \
  --data_dir data/input2 \
  --rnn_type lnlstm
```

```

INFO:dataloader:('checking:', 'data/input2')
INFO:dataloader:('loadtxt', 'data/input2/04.csv')
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 1, step 200, loss = 0.00993, elapsed = 0.795
epoch 1, step 400, loss = 0.00292, elapsed = 0.784
model saved to save3/model.ckpt-500
epoch 1, step 600, loss = 0.00165, elapsed = 0.805
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 2, step 800, loss = 0.00109, elapsed = 0.812
epoch 2, step 1000, loss = 0.00083, elapsed = 0.819
model saved to save3/model.ckpt-1000
epoch 2, step 1200, loss = 0.00078, elapsed = 0.821
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 3, step 1400, loss = 0.00047, elapsed = 0.781
model saved to save3/model.ckpt-1500
epoch 3, step 1600, loss = 0.00050, elapsed = 0.802
epoch 3, step 1800, loss = 0.00039, elapsed = 0.870
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 4, step 2000, loss = 0.00052, elapsed = 0.769
model saved to save3/model.ckpt-2000
epoch 4, step 2200, loss = 0.00028, elapsed = 0.791
epoch 4, step 2400, loss = 0.00033, elapsed = 0.797
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
model saved to save3/model.ckpt-2500
epoch 5, step 2600, loss = 0.00023, elapsed = 0.807
epoch 5, step 2800, loss = 0.00047, elapsed = 0.777
epoch 5, step 3000, loss = 0.00022, elapsed = 0.781
model saved to save3/model.ckpt-3000
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 6, step 3200, loss = 0.00023, elapsed = 0.809
epoch 6, step 3400, loss = 0.00020, elapsed = 0.797
model saved to save3/model.ckpt-3500
epoch 6, step 3600, loss = 0.00013, elapsed = 0.774
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
epoch 7, step 3800, loss = 0.00025, elapsed = 0.765
epoch 7, step 4000, loss = 0.00010, elapsed = 0.768
model saved to save3/model.ckpt-4000
epoch 7, step 4200, loss = 0.00019, elapsed = 0.887
INFO:dataloader:('num_batches:', 616, 'batch_size:', 50, 'seq_length:', 100)
^C

```

Traceback (most recent call last):

```

  File "train.py", line 154, in <module>
    train(args)
  File "train.py", line 110, in train
    feed)
  File "/opt/conda/envs/airi400/lib/python2.7/site-packages/tensorflow/python/client/session.py", line 895, in run
    run_metadata_ptr)
  File "/opt/conda/envs/airi400/lib/python2.7/site-packages/tensorflow/python/client/session.py", line 1124, in _run
    feed_dict_tensor, options, run_metadata)
  File "/opt/conda/envs/airi400/lib/python2.7/site-packages/tensorflow/python/client/session.py", line 1321, in _do_run
    options, run_metadata)
  File "/opt/conda/envs/airi400/lib/python2.7/site-packages/tensorflow/python/client/session.py", line 1327, in _do_call
    return fn(*args)
  File "/opt/conda/envs/airi400/lib/python2.7/site-packages/tensorflow/python/client/session.py", line 1306, in _run_fn
    status, run_metadata)
KeyboardInterrupt

```

```
In [67]: !python sample.py \
--save_dir save3 \
--motion_id 0,0,0,0,1 \
--output_file data/tmp/sample_04-3.csv
```

```
!python sample.py \
--save_dir save3 \
--motion_id 0,0,0,0,1 \
--output_file data/tmp/sample_04-3.csv
```

[illegible][illegible]

In [68]:

```
%%bash -e
python csv_normalize.py -r \
  --mean_file data/input2/mean.txt \
  --std_file data/input2/std.txt \
  data/tmp/sample_04-3.csv \
  data/tmp/__rel__.csv

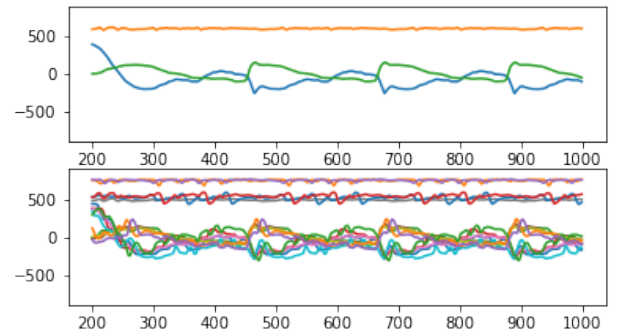
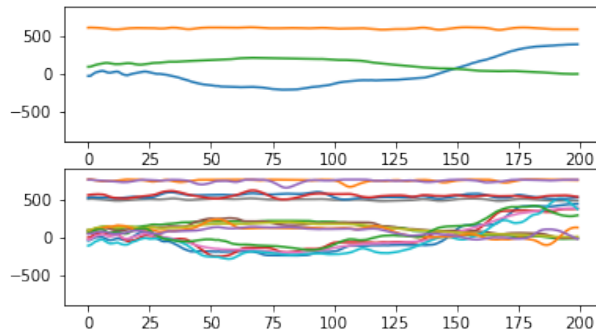
python csv_motion_to_jrel2.py -r \
  data/tmp/__rel__.csv \
  save3/sample_04-3.csv
```

```
{'scale': 1.0, 'std_file': 'data/input2/std.txt', 'reverse': True, 'input_file': 'data/tmp/sample_04-3.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input2/mean.txt', 'verbose': False}
un-normalize: data/tmp/sample_04-3.csv data/tmp/__rel__.csv
{'output_file': 'save3/sample_04-3.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
```



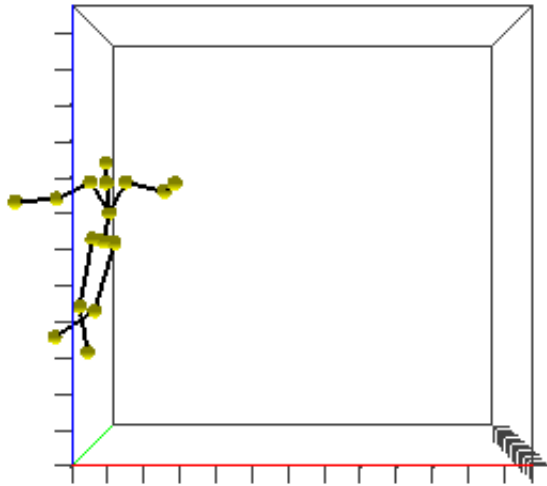
In [69]:

```
plot_motion('save3/sample_04-3.csv',  
            vlim=900.0)
```



In [70]:

```
show_motion('save3/sample_04-3.csv')
```



여러개 안무 동시 학습

In [65]:

```
!ls data/mocap-thkim-3d/
```

```
94_01_skeleton_3d.csv 94_05_skeleton_3d.csv 94_09_skeleton_3d.csv
94_02_skeleton_3d.csv 94_06_skeleton_3d.csv 94_10_skeleton_3d.csv
94_03_skeleton_3d.csv 94_07_skeleton_3d.csv 94_11_skeleton_3d.csv
94_04_skeleton_3d.csv 94_08_skeleton_3d.csv 94_12_skeleton_3d.csv
```

In [66]:

```
%%bash
for n in 01 02 03 04      # 05 06 07 08 09 10 11 12
do
    python csv_motion_to_jrel2.py \
        data/mocap-thkim-3d/94_${n}_skeleton_3d.csv \
        data/tmp/${n}-4.csv
done

python csv_stats.py \
    --mean_file data/input4/mean.txt \
    --std_file data/input4/std.txt \
    data/tmp/*-4.csv

for n in 01 02 03 04      # 05 06 07 08 09 10 11 12
do
    python csv_normalize.py \
        --mean_file data/input4/mean.txt \
        --std_file data/input4/std.txt \
        data/tmp/${n}-4.csv \
        data/input4/${n}.csv
done
```

```

{'output_file': 'data/tmp/01-4.csv', 'reverse': False, 'input_file': 'data/mocap-thkim-3d/94_01_skeleton_3d.csv'}
{'output_file': 'data/tmp/02-4.csv', 'reverse': False, 'input_file': 'data/mocap-thkim-3d/94_02_skeleton_3d.csv'}
{'output_file': 'data/tmp/03-4.csv', 'reverse': False, 'input_file': 'data/mocap-thkim-3d/94_03_skeleton_3d.csv'}
{'output_file': 'data/tmp/04-4.csv', 'reverse': False, 'input_file': 'data/mocap-thkim-3d/94_04_skeleton_3d.csv'}
{'var_file': None, 'verbose': False, 'std_file': 'data/input4/std.txt', 'mean_file': 'data/input4/mean.txt', 'input_files': ['data/tmp/01-4.csv', 'data/tmp/02-4.csv', 'data/tmp/03-4.csv', 'data/tmp/04-4.csv']}
input: data/tmp/01-4.csv
input: data/tmp/02-4.csv
input: data/tmp/03-4.csv
input: data/tmp/04-4.csv
wrote: data/input4/mean.txt
wrote: data/input4/std.txt
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': False, 'input_file': 'data/tmp/01-4.csv', 'output_file': 'data/input4/01.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
normalize: data/tmp/01-4.csv data/input4/01.csv
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': False, 'input_file': 'data/tmp/02-4.csv', 'output_file': 'data/input4/02.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
normalize: data/tmp/02-4.csv data/input4/02.csv
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': False, 'input_file': 'data/tmp/03-4.csv', 'output_file': 'data/input4/03.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
normalize: data/tmp/03-4.csv data/input4/03.csv
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': False, 'input_file': 'data/tmp/04-4.csv', 'output_file': 'data/input4/04.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
normalize: data/tmp/04-4.csv data/input4/04.csv

```

In [ ]:

```

!python -u train.py N
--save_dir save4 \
--data_dir data/input4 \
--rnn_type lnlstm \
--keep_prob 0.8 \
--num_epochs 20

```

## 1 ~ 4 번 동작 생성

In [103]:

```
%%bash
set -e

for n in 1 2 3 4
do
    python sample.py \
        --save_dir save4 \
        --motion_id $n \
        --output_file data/tmp/samples-$n.csv

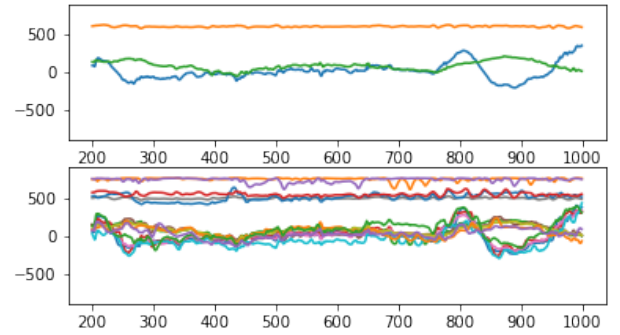
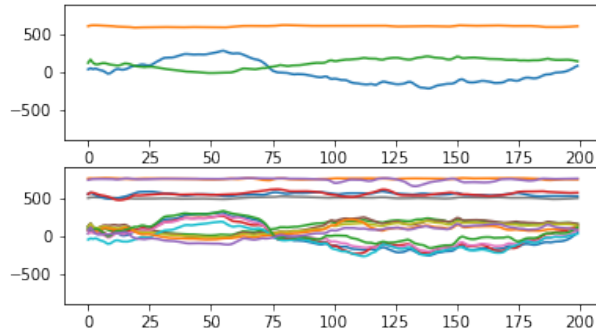
    python csv_normalize.py -r \
        --mean_file data/input4/mean.txt \
        --std_file data/input4/std.txt \
        data/tmp/samples-$n.csv \
        data/tmp/__rel__.csv

    python csv_motion_to_jrel2.py -r \
        data/tmp/__rel__.csv \
        save4/samples-$n.csv
done
```

```
loading model: save4/model.ckpt-27370
100%|#####| 1000/1000 [00:02<00:00, 400.96it/s]
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': True, 'input_file': 'data/tmp/samples-1.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
un-normalize: data/tmp/samples-1.csv data/tmp/__rel__.csv
{'output_file': 'save4/samples-1.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
loading model: save4/model.ckpt-27370
100%|#####| 1000/1000 [00:02<00:00, 383.85it/s]
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': True, 'input_file': 'data/tmp/samples-2.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
un-normalize: data/tmp/samples-2.csv data/tmp/__rel__.csv
{'output_file': 'save4/samples-2.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
loading model: save4/model.ckpt-27370
100%|#####| 1000/1000 [00:02<00:00, 384.57it/s]
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': True, 'input_file': 'data/tmp/samples-3.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
un-normalize: data/tmp/samples-3.csv data/tmp/__rel__.csv
{'output_file': 'save4/samples-3.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
loading model: save4/model.ckpt-27370
100%|#####| 1000/1000 [00:02<00:00, 387.19it/s]
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': True, 'input_file': 'data/tmp/samples-4.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
un-normalize: data/tmp/samples-4.csv data/tmp/__rel__.csv
{'output_file': 'save4/samples-4.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
```

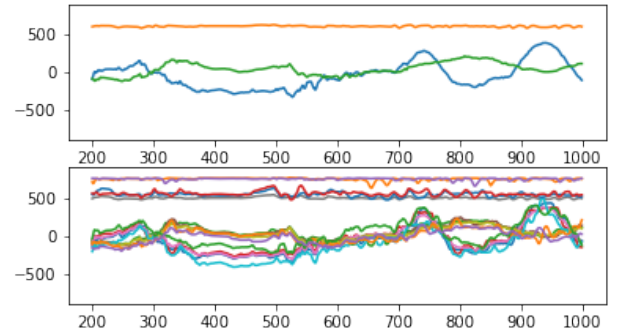
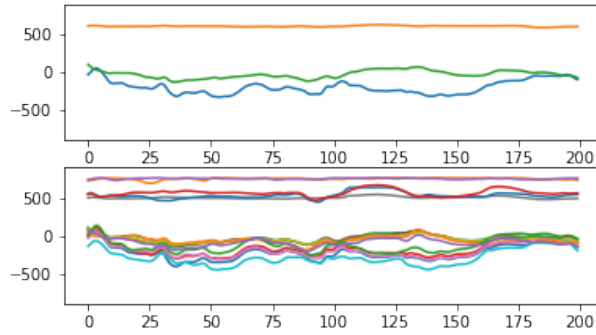
In [104]:

```
plot_motion('save4/samples-1.csv', vlim=900.0)
```



In [105]:

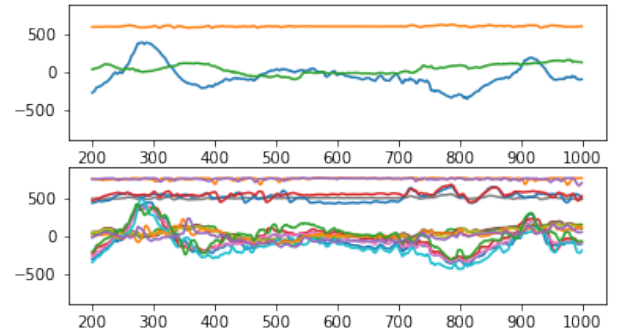
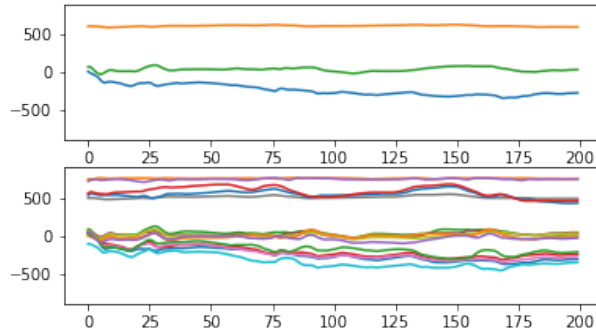
```
plot_motion('save4/samples-2.csv', vlim=900.0)
```





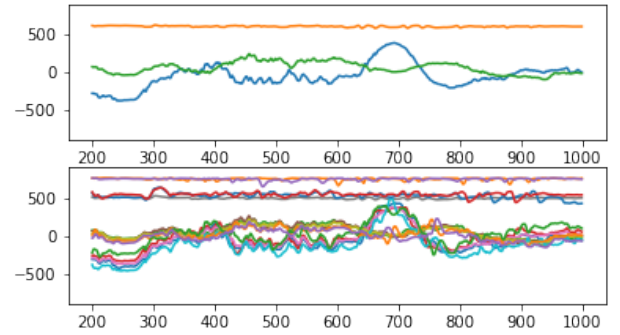
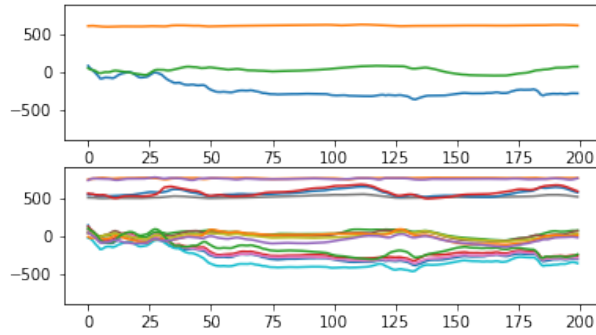
In [106]:

```
plot_motion('save4/samples-3.csv', vlim=900.0)
```



In [107]:

```
plot_motion('save4/samples-4.csv', vlim=900.0)
```



새로운 동작 테스트

In [108]:

```
%%bash -e
python sample.py \
  --save_dir save4 \
  --motion_id 0,0,0,1,1 \
  --output_file data/tmp/samples-3_4.csv

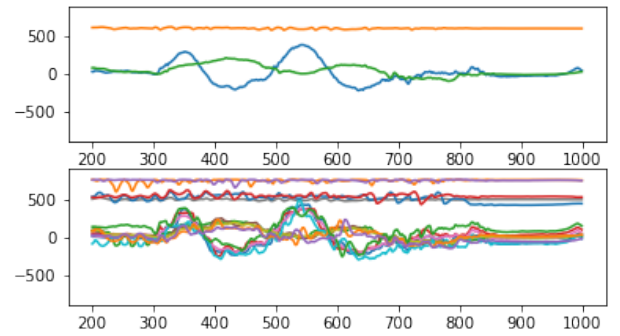
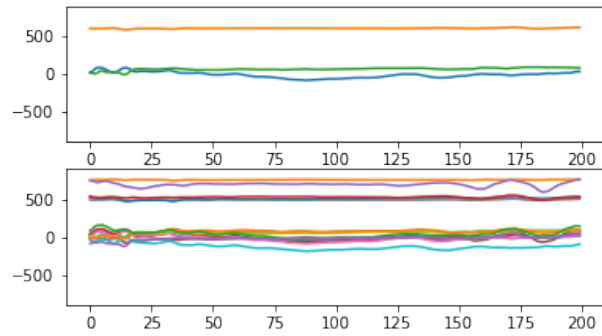
python csv_normalize.py -r \
  --mean_file data/input4/mean.txt \
  --std_file data/input4/std.txt \
  data/tmp/samples-3_4.csv \
  data/tmp/__rel__.csv

python csv_motion_to_jrel2.py -r \
  data/tmp/__rel__.csv \
  save4/samples-3_4.csv
```

```
loading model: save4/model.ckpt-27370
100%|#####| 1000/1000 [00:02<00:00, 398.26it/s]
{'scale': 1.0, 'std_file': 'data/input4/std.txt', 'reverse': True, 'input_file': 'data/tmp/samples-3_4.csv', 'output_file': 'data/tmp/__rel__.csv', 'mean_file': 'data/input4/mean.txt', 'verbose': False}
un-normalize: data/tmp/samples-3_4.csv data/tmp/__rel__.csv
{'output_file': 'save4/samples-3_4.csv', 'reverse': True, 'input_file': 'data/tmp/__rel__.csv'}
```

In [109]:

```
plot_motion('save4/samples-3_4.csv',  
            vlim=900.0)
```



In [110]:

```
%%html
<a href="catalog-before-after.html" target="_">
click me!
</a>
```

click me!  
(catalog-before-after.html)

## 참고문헌

- “Generating Sequences With Recurrent Neural Networks”, Alex Graves, 2013 (arXiv:[1308.0850](https://arxiv.org/abs/1308.0850) (<https://arxiv.org/abs/1308.0850>)).
- “Generative Choreography using Deep Learning”, Luka Crnkovic-Friis, Louise Crnkovic-Friis, 2016 (arXiv:[1605.06921](https://arxiv.org/abs/1605.06921) (<https://arxiv.org/abs/1605.06921>)).
- <https://github.com/hardmaru/write-rnn-tensorflow> (<https://github.com/hardmaru/write-rnn-tensorflow>) ( [blog.otoro.net](http://blog.otoro.net) (<http://blog.otoro.net/2015/12/12/handwriting-generation-demo-in-tensorflow/>) )

