sentence_articulation

June 15, 2017

1 Sentence Articulation

1.1 Load Sentence

```
In [1]: import timit_stats as ts
        import gesture as ges
        import os
        import matplotlib.pyplot as plt

        %pylab inline

        root_dir = "../USC-TIMIT/EMA/Data/M1"
        index = 0

        t_names, m_names = zip(*ts.list_TIMIT_dir(root_dir))
        trans_fname = t_names[index]
        mat_fname = m_names[index]

        # parse .trans file
        t_starts, t_ends, phonemes, words, sentences = ts.parse_transcription(trans_fname)
        phones = list(set(phonemes))
        # parse .mat file
        params, srates = ts.parse_mat(mat_fname)
        print "Sentence \"{}\" loaded succesfully".format(trans_fname)
```

```
Populating the interactive namespace from numpy and matplotlib
Sentence "../USC-TIMIT/EMA/Data/M1\trans\usctimit_ema_m1_001_005.trans" loaded succesfully
```

1.2 Calc Gestures And Variances

```
In [2]: gestures = {}
        means = {} # key : param_name, value: dict(ges, val)
        variances = {} # key : param_name, value: dict(ges, val)

        articulators = ["LL", "UL", "TT", "TB", "TD", "JAW"]
        domains=["_x", "_y"]
        param_names = [a+d for a in articulators for d in domains]
```

```
        for i in range(len(t_names)):
            t_fname = t_names[i]
            mat_fname = m_names[i]
            gest = ts.calc_gestures(mat_fname, t_fname, filter_critical_points=True, m=0.05)
            for g in gest:
                if g not in gestures:
                    gestures[g] = ges.Gesture(g)
                gestures[g].extend(gest[g])
        print "gestures calculation finished"

        gestures_norm, p_max, p_min = ts.normalize_gestures(gestures)

        for p in param_names:
            means[p] = {}
            variances[p] = {}

        for g in gestures_norm:
            g_m = gestures_norm[g].get_mean()
            g_v = gestures_norm[g].get_variance()
            for p in param_names:
                means[p][g] = g_m[p]*(p_max[p]-p_min[p]) + p_min[p]
                variances[p][g] = g_v[p]

        print "Means and variances calculated succesfully"
```

```
critical_point.py:9: RuntimeWarning: invalid value encountered in double_scalars
  velocity = [math.sqrt(dx**2 + dy**2) for dx, dy in zip(delta_ax, delta_ay)]
critical_point.py:12: RuntimeWarning: invalid value encountered in double_scalars
  math.sqrt(delta_ax[i-1] ** 2 + delta_ay[i-1] ** 2))) for i in range(1, len(delta_ax))]


gestures calculation finished
Means and variances calculated succesfully
```

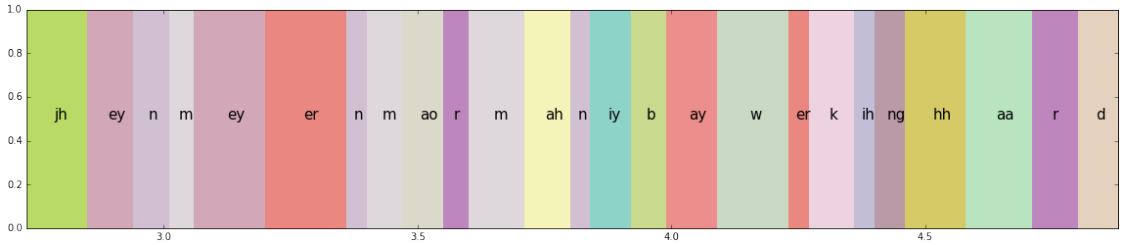
1.3 Find Critical Points

```
In [55]: import critical_point as cp
         m=0.01
         critical_points = {}
         for a in articulators:
             critical_points[a+"_x"] = cp.find_critical_points(a, params, m)
             critical_points[a+"_y"] = cp.find_critical_points(a, params, m)
```

1.4 Plot Sentence Phonemes

```
In [56]: import matplotlib.patches as patches
```

```
cmap = plt.get_cmap('Set3')
colors = [cmap(i) for i in np.linspace(0, 1, len(phones))]
colors = dict(zip(phones, colors))

# show first 20 phonemes except first one ("silence")
show_start = 15
num_to_show = 25
show_last = min(len(phonemes), show_start + num_to_show)


fig1, ax1 = plt.subplots(figsize=(20, 4))
ax1.set_xlim(t_starts[show_start], t_ends[show_last-1])
ax1.set_ylim(0, 1.0)
for i in range(show_start, show_last):
    width = t_ends[i]-t_starts[i]
    height = 1.0
    ax1.add_patch(patches.Rectangle((t_starts[i], 0.0),
                                    width, height, color=colors[phonemes[i]]))
    ax1.text(t_starts[i]+width/2-0.005, height/2, phonemes[i], fontsize=15)
plt.show()
```

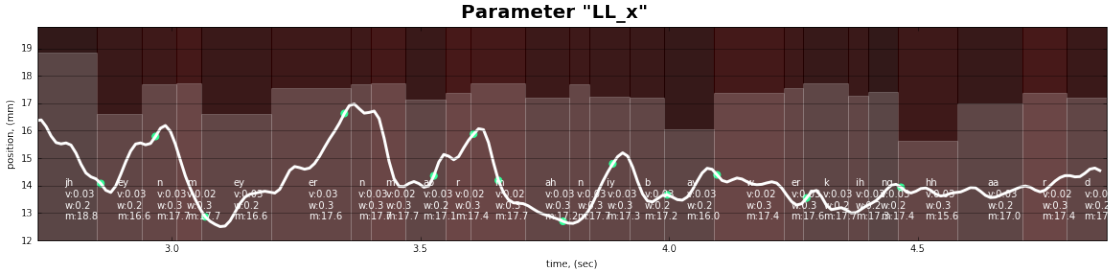


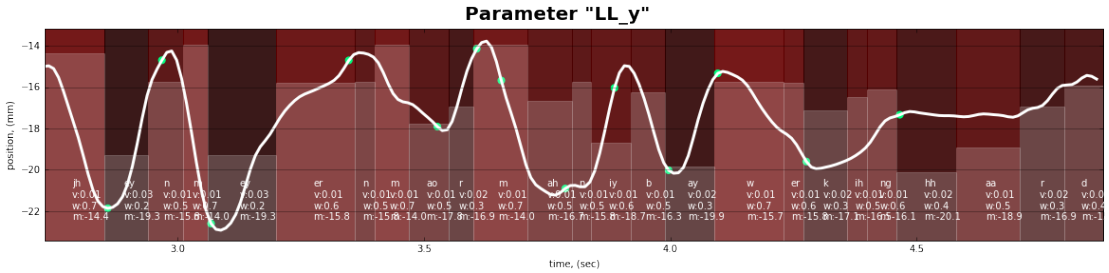
Plot Articulators

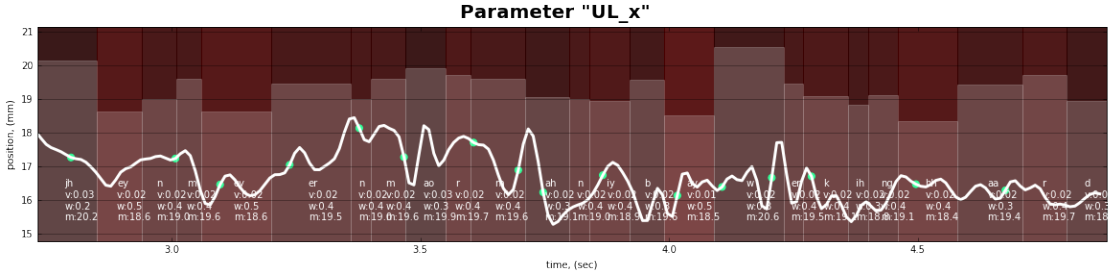
In [57]:
```
import copy

cmap = plt.get_cmap('gist_heat')
colors = [cmap(i) for i in range(100)]

for p in param_names:
# get min and max variance and means for p
    v_min = min(variances[p].itervalues())
    v_max = max(variances[p].itervalues())
    v_range = v_max - v_min
    m_min = min(means[p].itervalues())
    m_max = max(means[p].itervalues())
# prepare figure
    fig1, ax = plt.subplots(figsize=(20, 4))
```

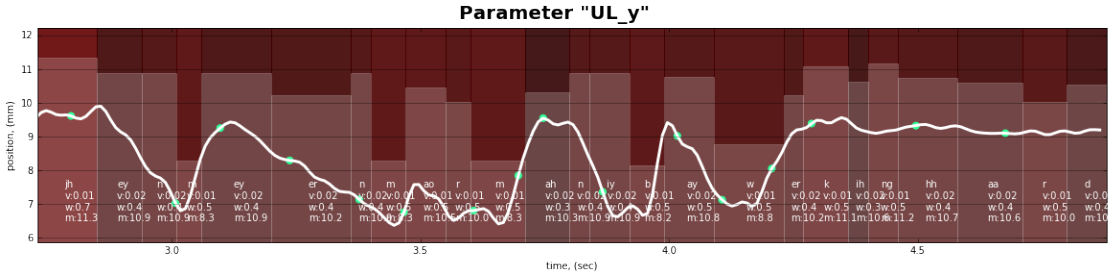
```
    fig1.suptitle("Parameter \"{}\"".format(p),
                  fontsize=20, fontweight='bold')
    ax.grid(color='black', linestyle='-', linewidth=1, alpha=0.3, axis='y')
    ax.set_xlabel("time, (sec)")
    ax.set_ylabel("position, (mm)")
# calc range
    ax.set_xlim(t_starts[show_start], t_ends[show_last-1])

    rate = srates[p]
    i_start = int(t_starts[show_start] * rate)
    i_end = int(t_ends[show_last-1] * rate)
    length = i_end - i_start
    y = params[p][i_start:i_end]
    p_min = min(min(y), m_min) - 0.5
    p_max = max(max(y), m_max) + 0.5

    ax.set_ylim(p_min, p_max)
    height = p_max - p_min

    for i in range(show_start, show_last):
        phone = phonemes[i]
        v = variances[p][phone]
        varian = gestures_norm[phone].get_variance()
        w = exp(-50*v) # varian[p]/ sum(varian.values()))
        #       w is weight of parameter for this gesture
#         w = 1 - (gestures_norm[phone].get_variance[p] v-v_min)/(v_max-v_min)
        m = means[p][phone]
        clr = colors[int(w*100)]

        width = t_ends[i]-t_starts[i]
        ax.add_patch(patches.Rectangle((t_starts[i], p_min),
                                       width, height, color=clr, alpha=0.9, zorder=0))
        ax.add_patch(patches.Rectangle((t_starts[i], p_min),
                                       width, m - p_min, color="w", alpha=0.2))
        text1 =  "{0}\nv:{1:0.2f}\nw:{2:0.1f}\nm:{3:0.1f}".format(phonemes[i], v, w, m
        ax.text(t_starts[i]+width/2-0.005,
                p_min+height/10, text1, fontsize=10, color='w')


    t = range(i_start,i_end)
    t = [i / srates[p] for i in t]
    crit_points_i = [i for i in critical_points[p] if i >= i_start and i <= i_end]
    crit_y = [params[p][i] for i in crit_points_i]
    crit_t = [i/srates[p] for i in crit_points_i]
    ax.plot(t, y, color="w", linewidth=3)
    ax.scatter(crit_t, crit_y, s=[50]*len(crit_y), color='springgreen', alpha=1)
```

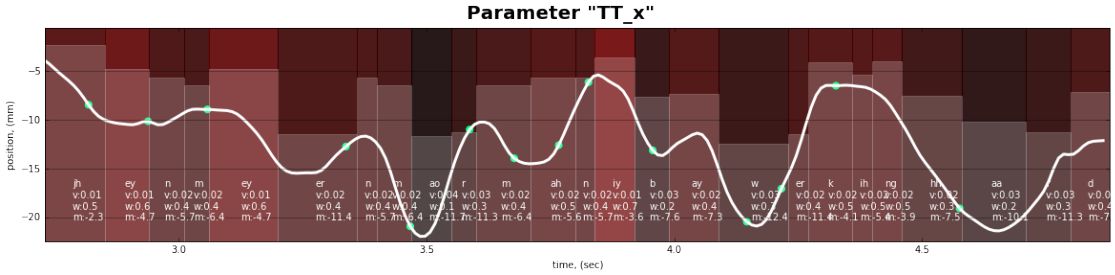
```
plt.show()
```

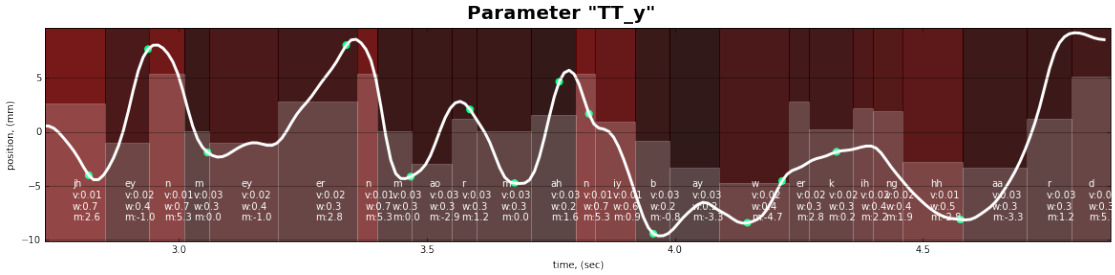

Parameter "LL_x"
position, (mm)
time, (sec)

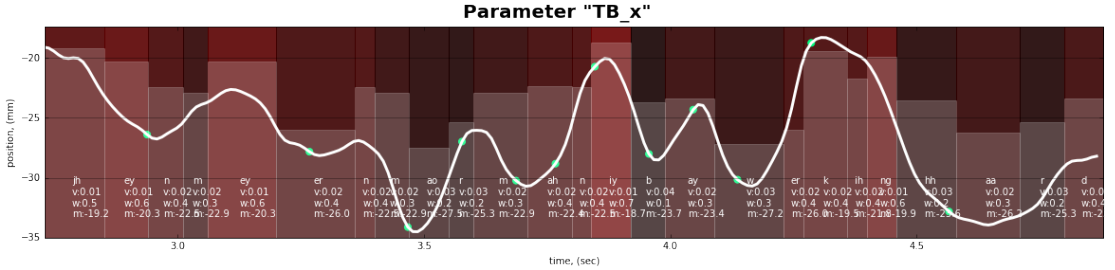

Parameter "LL_y"
position, (mm)
time, (sec)

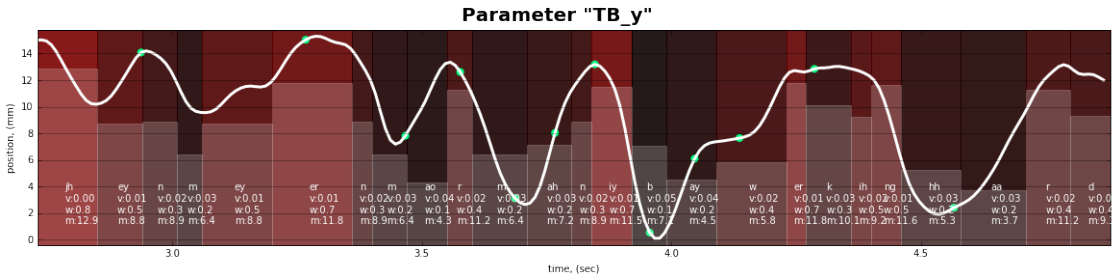

Parameter "UL_x"
position, (mm)
time, (sec)

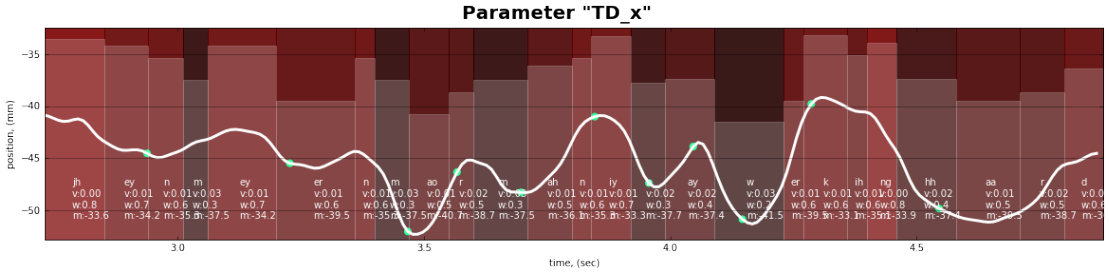

Parameter "UL_y"
position, (mm)
time, (sec)

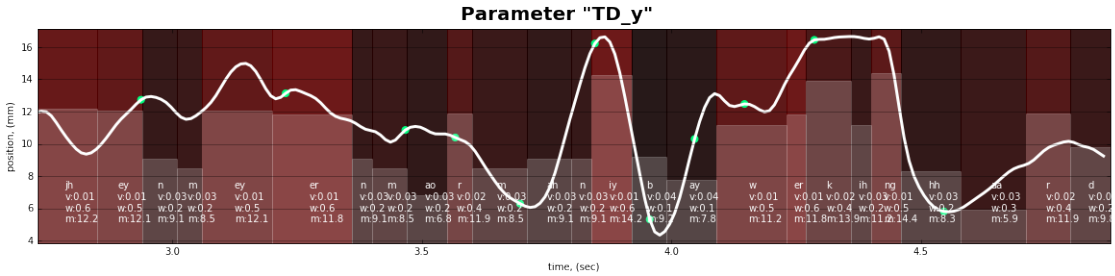

Parameter "TT_x"
position, (mm)
time, (sec)

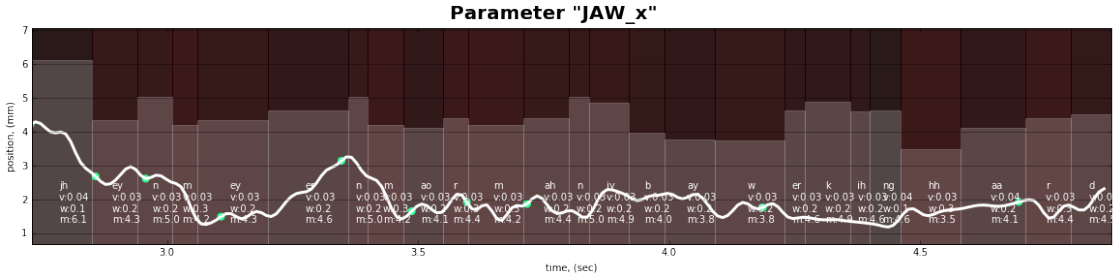

Parameter "TT_y"
position, (mm)
time, (sec)

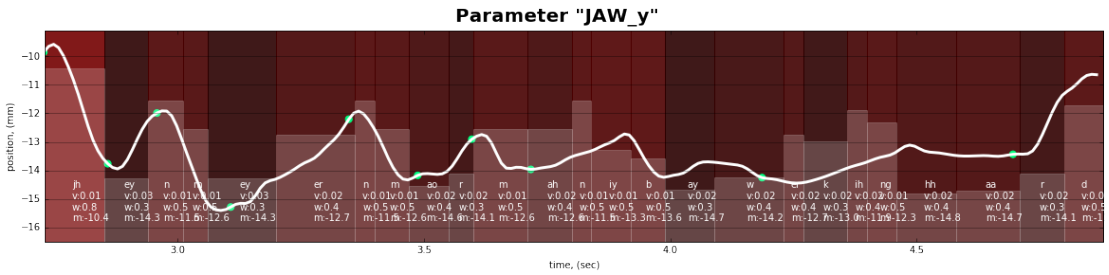

Parameter "TB_x"
position, (mm)
time, (sec)


Parameter "TB_y"
position, (mm)
time, (sec)


Parameter "TD_x"
position, (mm)
time, (sec)


Parameter "TD_y"
position, (mm)
time, (sec)


Parameter "JAW_x"
position, (mm)
time, (sec)


Parameter "JAW_y"
position, (mm)
time, (sec)

In [ ]:

In [ ]:

In [ ]: