General rules:

- For all figures that you generate, remember to add meaningful labels to the axes, and make a legend, if applicable.
- Do not hard code constants, like number of samples, number of channels, etc in your program. These values should always be determined from the given data. This way, you can easily use the code to analyse other data sets.
- Do not use high-level functions from toolboxes like scikit-learn.
- Replace *Template* by your *FirstnameLastname* in the filename, or by *Lastname1Lastname2* if you work in pairs.

BCI-IL - Exercise Sheet #05

Name

In [10]:

```
% matplotlib inline

import numpy as np
import scipy as sp
import scipy.signal
from matplotlib import pyplot as plt


import bci_minitoolbox as bci
```

Preparation: Load data

In [11]:

```
fname= 'eyes_closed_VPal.npz'
cnt, fs, clab, mnt =bci.load_data(fname)
print(cnt.shape) #118x5958
print(len(clab)) #118
print(mnt.shape) #118x2
```

```
(118, 5958)
118
(118, 2)
```

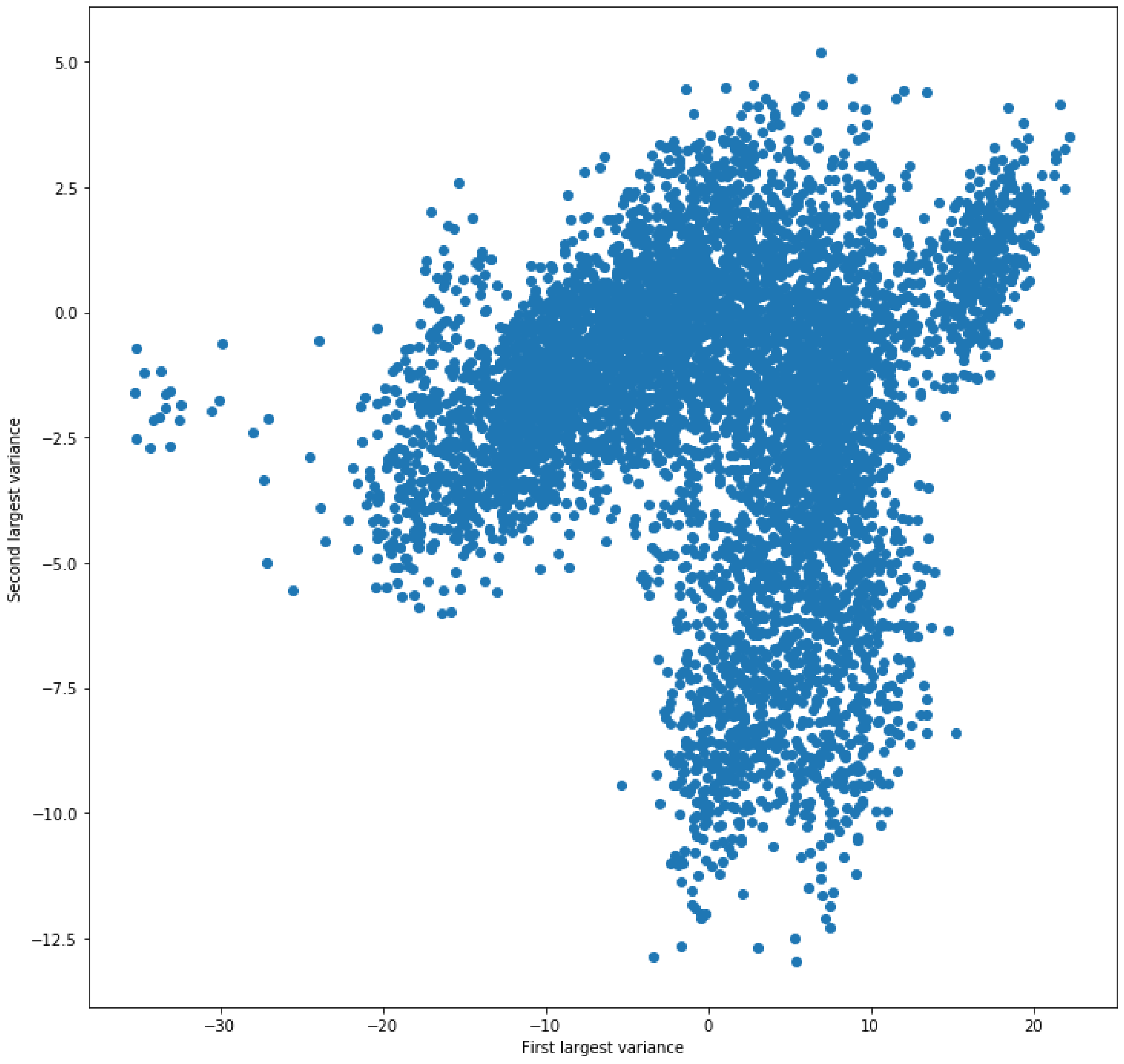
Exercise 1: PCA on raw data (3 points)

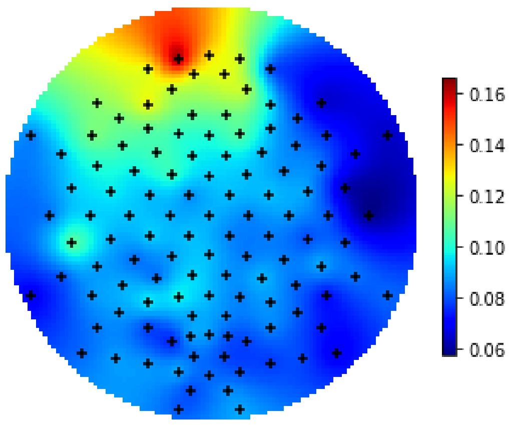
Make a scatter plot of the data with the two directions of largest variance as coordinate axes. Then, depcit the projection vectors of those two components as scalp maps (function `scalpmap` provided in the `bbci_minitoolbox`).

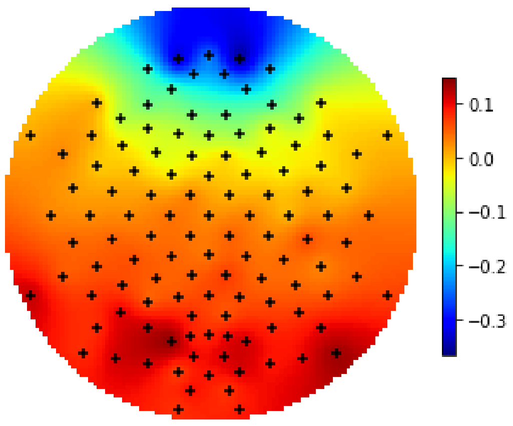
In [26]:

```
X = (cnt - np.mean(cnt))/np.std(cnt)       # Standardization of variables as pre
-processing step



C = np.cov(X)
print(cnt.shape)
print(C.shape)
eig_vals, eig_vecs =np.linalg.eigh(C)
print (eig_vals.shape)
print(eig_vecs.shape)

# Shae the two principal axes as a vector for
matrix_w = np.hstack((eig_vecs[:,-1].reshape(len(eig_vecs),1),eig_vecs[:,-2].res
hape(len(eig_vecs),1)))

print ( np.shape(matrix_w))

#Transform the data to the principal axes co-ordinate system
Y = X.T.dot(matrix_w)

print ( np.shape (Y))

plt.figure(figsize = (12,12))
# plt.scatter(eig_vecs[:,-1], eig_vecs[:,-2])
plt.scatter(Y[:,0], Y[:,1])
plt.xlabel("First largest variance")
plt.ylabel("Second largest variance")

plt.figure(figsize = (12,6))
plt.subplot(121)
bci.scalpmap(mnt,eig_vecs[:,-1])
plt.subplot(122)
bci.scalpmap(mnt,eig_vecs[:,-2])
```

```
(118, 5958)
(118, 118)
(118,)
(118, 118)
(118, 2)
(5958, 2)
```







Exercise 2: Artifact to signal ratio with PCA (5 points)

For this task we assume that the two components from Ex. #01 reflect eye movements, while all other components do not contain artifacts from eye movement. If you did not succeed with Ex. #01, chose an arbitrary component.

Determine for each channel which proportion of the overall variance is caused by eye movements and plot this information as a scalp map. Also, calculate the Signal-To-Noise ratio (SNR) per channel in Decibel (dB).

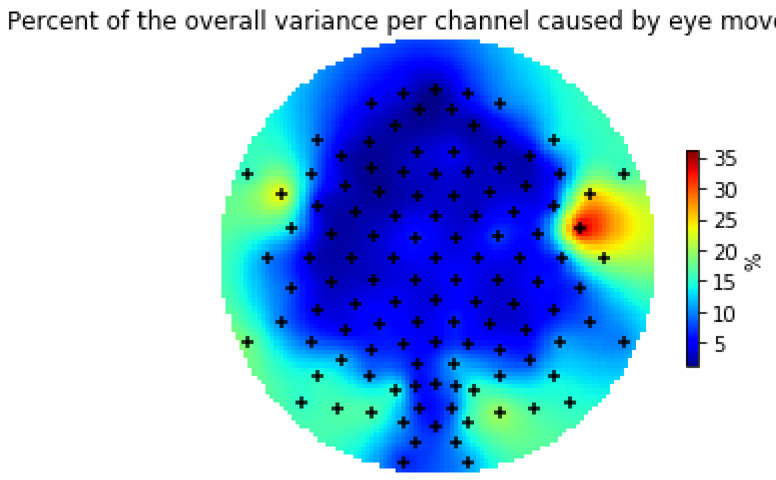
In [4]:

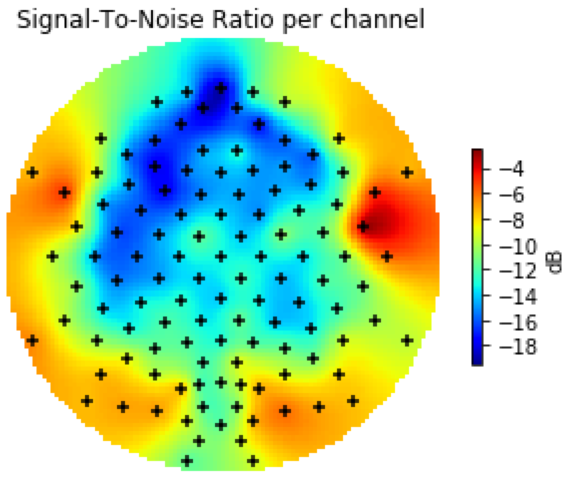
```
idx=[-1,-2]
cnt_s = V[:, idx].T@cnt # W.TX
print(cnt_s.shape)

x_eyes=V[:,idx]@cnt_s #AS
print(x_eyes.shape)

cnt_artifree=cnt-x_eyes
C_clean = np.cov(cnt_artifree)
C_noise = np.cov(x_eyes)

a = np.diagonal(C).copy()
b = np.diagonal(C_clean).copy()
c = np.diagonal(C_noise).copy()

proportion = np.divide(b,a)
SNR = np.divide(b,c)
print(SNR)
SNR_db = 10 * np.log10(SNR)
print(SNR_db)

plt.figure()
bci.scalpmap(mnt, proportion*100, cb_label = '%')
plt.title('Percent of the overall variance per channel caused by eye movements')

plt.figure()
bci.scalpmap(mnt,SNR_db,cb_label = 'dB')
plt.title('Signal-To-Noise Ratio per channel')
```

```
(2, 5958)
(118, 5958)
[0.04117148 0.0125623  0.01103606 0.02875167 0.06670432 0.04908415
 0.02053914 0.01766199 0.10575103 0.02321465 0.03645448 0.0551194
 0.02710009 0.10572214 0.02903875 0.01586516 0.03044617 0.03476235
 0.03316477 0.0345375  0.02536871 0.1546038  0.11457591 0.03407083
 0.01669189 0.03008247 0.03476775 0.03476789 0.03257575 0.06790173
 0.1815352  0.33189666 0.03324921 0.02544456 0.03201951 0.042952
 0.03113454 0.05727149 0.06628821 0.34167808 0.18550278 0.15053264
 0.02333207 0.0350121  0.07335829 0.0384789  0.08883425 0.06889233
 0.56887516 0.10680973 0.02950028 0.02852497 0.04389717 0.04682786
 0.04598268 0.04243431 0.07489315 0.26481501 0.09139603 0.02883172
 0.03315253 0.04121504 0.06066977 0.03720396 0.06680252 0.07801678
 0.22530586 0.10383847 0.04050055 0.04997187 0.05346817 0.03897511
 0.04586196 0.03832836 0.05847714 0.09586087 0.10841644 0.09057888
 0.04700158 0.06498477 0.04470261 0.07471549 0.03813472 0.04294965
 0.12570646 0.1779728  0.18213524 0.13553075 0.0691771  0.06446931
 0.04922145 0.0608588  0.07228524 0.07409783 0.16023814 0.17666462
 0.20288876 0.10206659 0.06081616 0.13406612 0.11794077 0.19371778
 0.20680319 0.18393664 0.05889886 0.07807396 0.11863616 0.1937559
 0.25848674 0.06202598 0.08589326 0.11991543 0.06443613 0.19104273
 0.08932338 0.1246812  0.06744156 0.11992543]
[-13.85403486 -19.00930703 -19.57185846 -15.41336999 -11.75846036
 -13.09058683 -16.8741766  -17.52960244  -9.75715408 -16.34237904
 -14.38249058 -12.58695514 -15.67029274  -9.75834061 -15.37022053
 -17.99555586 -15.16467342 -14.58890885 -14.79323024 -14.61709073
 -15.95701614  -8.10779841  -9.40906683 -14.67617279 -17.77494471
 -15.21686527 -14.58823384 -14.58821649 -14.87105542 -11.68119153
  -7.41039149  -4.78997112 -14.78218616 -15.94405102 -14.94585325
 -13.67016636 -15.06757597 -12.42061524 -11.78563722  -4.66382882
  -7.31649576  -8.22369318 -16.32046658 -14.55781819 -11.34550818
 -14.14777303 -10.5141955  -11.61829149  -2.4498303   -9.71389194
 -15.30173924 -15.4477476  -13.57563485 -13.29495719 -13.37405709
 -13.72282829 -11.25557902  -5.77057397 -10.39072662 -15.40129467
 -14.79483268 -13.84944311 -12.17027635 -14.29410882 -11.75207171
 -11.07811974  -6.47227514  -9.83641716 -13.9253909  -13.01274437
 -12.71904716 -14.09212658 -13.38547343 -14.16479797 -12.3301391
 -10.18358639  -9.64904856 -10.42973052 -13.27887555 -11.87188392
 -13.49667117 -11.26589354 -14.18679398 -13.67040361  -9.0064242
  -7.49646373  -7.39606025  -8.6796216  -11.60037625 -11.90646983
 -13.07845565 -12.15676594 -11.40950394 -11.30194528  -7.95234096
  -7.52850422  -6.92742018  -9.91116384 -12.1598102   -8.72680967
  -9.28336028  -7.12830528  -6.8444277   -7.35331756 -12.29893119
 -11.07493819  -9.25782924  -7.12745062  -5.8756173  -12.07426371
 -10.66040894  -9.21124929 -11.90870557  -7.18869489 -10.4903487
  -9.04199012 -11.71072376  -9.21088721]
```

Out[4]:

```
Text(0.5,1,'Signal-To-Noise Ratio per channel')
```





Preparation: Load data

In [37]:

```
fname = 'erp_hexVPsag.npz'
cnt, fs, clab, mnt, mrk_pos, mrk_class, mrk_className = bci.load_data(fname)
```

```
(55, 2)
```

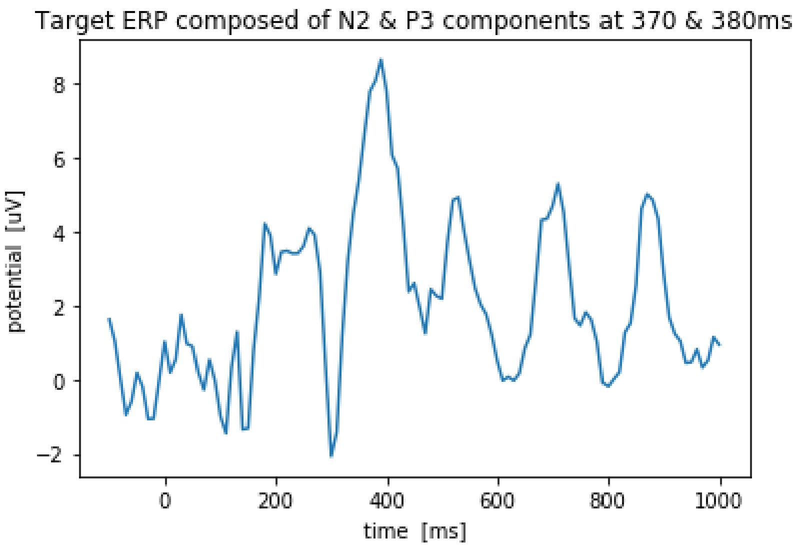
Exercise 3: Artificial EEG data (7 points)

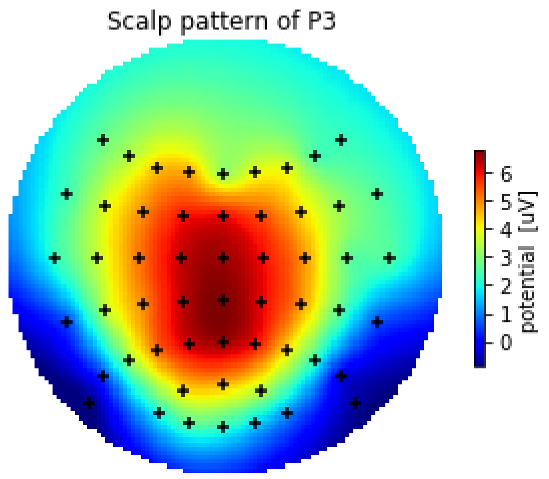
Generate one trial of artificial, stereotypical EEG data (1000 ms, 55 channels) out of the data set of sheet #01. The trial should contain a 'clean' target ERP composed of an N2 component (the one negatively peaking at 310 ms in the data on sheet #01) and a P3 component (the one peaking at 380 ms in the data on sheet #01). Both components should have their typical spatial distribution. To this extent, extract the corresponding scalp patterns at the peaks of the average ERPs, calculate the filters, use them to isolate the components from the average ERP and then project them back into the EEG space. Plot the artificial EEG (the backprojected ERP) in channels PO7 and Cz and the scalp patterns correpsonding to the N2 and P3.

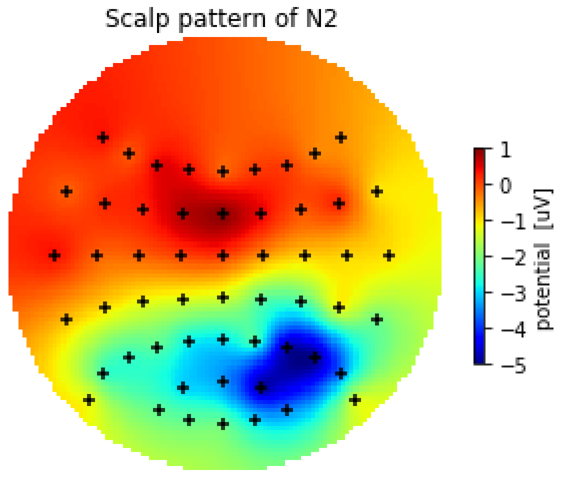
In [126]:

```
ival= [-100, 1000]
ref_ival= [-100, 0]

# Segment continuous data into epochs:
epo, epo_t = bci.makeepochs(cnt, fs, mrk_pos, ival)
# Baseline correction:
epo = bci.baseline(epo, epo_t, ref_ival)

erp = np.mean(epo[:, :, mrk_class==0], axis=2)

erp_1st_plot = erp[:,clab.index('Cz')] + erp[:,clab.index('PO7')]
maxx = np.where(erp_1st_plot == erp_1st_plot.max())[0][0]
minn = np.where(erp_1st_plot == erp_1st_plot.min())[0][0]

plt.figure()
plt.plot(epo_t, erp_1st_plot)
plt.title('Target ERP composed of N2 & P3 components at 370 & 380ms')
plt.xlabel('time  [ms]')
plt.ylabel('potential  [uV]')

plt.figure()
bci.scalpmap(mnt,erp[maxx,:] , cb_label = 'potential  [uV]')
plt.title('Scalp pattern of P3')

plt.figure()
bci.scalpmap(mnt,erp[minn,:] , cb_label = 'potential  [uV]')
plt.title('Scalp pattern of N2')



## calculate the filters, use them to isolate the components from the average ER
P and then project them back into the EEG space.

# no idea what to do here at all, didnt quite get the whole concept of filters a
nd there is no available information
# about that from the slides, so no idea how to proceed further
```

```
Out[126]:

Text(0.5,1,'Scalp pattern of N2')
```







In [ ]:

```
##
```

In [ ]:

```
##
```