# How Tranquility Works

**tomas.ukkonen@iki.fi, 2017**

Tranquility is experimental software that uses EEG-measurements and machine learning to show pictures and play sounds that aims to push and keep brain in a target EEG-state such as a meditative (strong alpha frequencies) or relaxed state. The software uses new cheap customer-grade EEG-devices (currently only Interaxon Muse is supported) to measure brain states. The motivation to develop such software comes from the frustration of psychiatric treatments that has been stuck to medication based approach. Patients (and their brain functions) are seldom or if not-at-all (continuosly) measured, medications are static interventions that do not respond dynamically to what is happening inside brain and there is no aim to use gentler approaches which would have have less side effects (such as audiovisual stimulation). If properly done, audiovisual stimulation has potential to cause wide range of emotional responses (think horror movies) and have lasting effect on brain (if you see something interesting or important you remember it for a long time).

Tranquility works by first showing and playing semirandom stimulation to brain and by using machine learning (artificial neural networks) to learn how different sounds and pictures change brain state. Due to the limited number of measurement points in customer-grade EEG-devices, there is no attempt to model how responses differ in different parts of the brain and only average frequency power of delta, theta, alpha, beta, gamma bands etc. under the whole area of the brain is measured as a single brain state vector $\mathbf{s}$. The measurements $\boldsymbol{s}$ are, however, processed using Hidden Markov Models (HMM) by discretizing them (k-means clustering) and using HMM-algorithm in attempt to learn deeper brain states.

After the initial learning phase, Tranquility software uses non-linear reinforcement learning extended to continuous actions and states. Mahalobinis distance to target brain state $\boldsymbol{t}$ is used as the reinforcement signal $r(\boldsymbol{s}) = (\boldsymbol{s} - \boldsymbol{t})^T \boldsymbol{D}^{-1} (\boldsymbol{s} - \boldsymbol{t})$ that aims to keep distance to $\boldsymbol{s}$ as small as possible. For pictures $\boldsymbol{a}_p$ (continuous picture vector), the reinforcement learning uses a Q-learning artificial neural network $Q(\boldsymbol{s}, \boldsymbol{a}_p)$ to find the best picture (from discrete set of user suplied pictures) to minimizes expected future distances to the target state $\boldsymbol{s}$. For sounds, continuous FM sound synthesizer's parameters $\boldsymbol{a}_f$ are optimized (*Continuous Control with Deep Reinforcement Learning, Google Deepmind, 2016*) by learning policy function $\boldsymbol{\mu}$ (artificial neural network) that maps response from current state $\boldsymbol{s}$ to continous FM sound synthesizer action $\boldsymbol{a}_f$. The policy is optimized by using policy gradient $\nabla_{\boldsymbol{w}} J = E_{\boldsymbol{s}_t}[\nabla_{\boldsymbol{w}} Q(\boldsymbol{s} = \boldsymbol{s}_t, \boldsymbol{a}_f = \boldsymbol{\mu}(\boldsymbol{s}_t|\boldsymbol{w}))]$.

## Future work

Currently, results are not very good but hand-tuning many different meta-parameters of neural network, hidden markov models and reinforcement learning may be able to improve results significantly.

There was attempt to extend sound synthesis based stimulation to computer-based composition of music using RNN-RBM and MIDI notes database but I ran out of computer power to compute sufficiently detailed model that would give good enough results (results that would sound like music).

Currently, the picture based stimulation uses a discrete set of preselected pictures and normal feedforward neural network operating on thumbnails of full-sized pictures. Furtherwork would modify traditional feedforward neural network to convolution neural network (ConvNet) with weight sharing so that full-sized pictures could be processed and neural network could learn to process fine details of the pictures. Additionally, if there was *much more* RAM, policy-based method could be extended to pictures and would allow using continuous picture stimulation instead of preselected pictures.

Other work could include the use of deep learning (stacked RBMs weight prelearning) and more complicated non-linear differential equations model for learning and stimulating responses to different stimuli. However, the real-time computation of even more complicated models would require much more powerful computer than what is currently available for home users. Another possible way to increase sophistication would be modelling multipoint brain measurement results and solving inverse problem to have access to true deep brain states instead of treating brain as a black box with no knowledge of its internal function.

The size of software (including my dinrhiw support library) is currently 90.000 lines of C++ code and small Java user interface (SWT) for C++ JNI module.