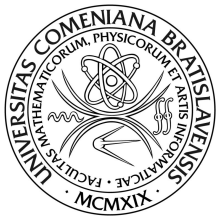


Katedra Aplikovanej Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského v Bratislave

Využitie rozhrania mozog–počítač na vykonávanie akcií

Diplomová práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: doc. Ing. Igor Farkaš, PhD.

Bratislava, 2013 Bc. Martin Kokoška

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Kokoška
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Využitie rozhrania mozog-počítač na vykonávanie akcií

Cieľ: S využitím meracieho zariadenia EEG implementujte učiaci systém s rozhraním mozog-počítač (BCI), ktorý bude schopný využiť spätnú väzbu na realizáciu akcií pomocou mysle. K tomu budete potrebovať:
1. naštudovať si literatúru z oblasti BCI systémov (metódy merania EEG, algoritmy spracovania a vyhodnocovania signálu),
2. zoznámiť sa s meracím zariadením EEG (g.USBamp od firmy g.tec je k dispozícii),
3. získať praktické skúsenosti s meraním,
4. otestovať funkčnosť systému na jednoduchšej úlohe (napr. ovládanie kurzora myši na obrazovke).

Poznámka: Požiadavky: záujem o danú problematiku, angličtina, relatívna samostatnosť, ochota pracovať priebežne.

Vedúci: doc. Ing. Igor Farkaš, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 13.10.2010

Dátum schválenia: 19.11.2010 doc. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Chcel by som poďakovať môjmu školiteľovi doc. Ing. Igorovi Farkašovi, PhD. za množstvo času, ktoré mne a tejto práci venoval a za všetky cenné rady a vedomosti, ktoré mi odovzdal.

Ďalej by som chcel poďakovať RNDr. Barbore Cimrovej, PhD., za hodiny strávené pri meraniach a konzultáciach a za objav, že predstava plutvového pohybu nohami je najlepším stavom relaxu.

Abstrakt

V našej práci sme používali rozhranie mozog–počítač na ovládanie guličky smerom k cieľu v 1D priestore pomocou aplikácie BCI2000. Vyhodnocovanie elektrickej aktivity mozgu (EEG) bolo založené na motorickej predstavivosti detekovanej pomocou zmien spektrálnej hustoty v tzv. μ-rytme v oblasti motorickej kôry. Na základe najväčších rozdielov v μ-rytme boli z nameraného EEG signálu offline metódou extrahované príznaky, ktoré boli vstupom do klasifikátora. Zamerali sme sa na adaptívne metódy klasifikácie týchto príznakov a nahradili sme pôvodný klasifikátor učiaci sa *s učiteľom* na klasifikátor založený na lineárnej diskriminačnej analýze, ktorý sa dokázal adaptovať aj *bez učiteľa*. Opakovanými meraniami na štyroch subjektoch sme vyhodnocovali úspešnosť rôznych typov adaptácie klasifikátora. Lepšie výsledky ako pôvodný klasifikátor implementovaný v BCI2000 dosiahol LDA klasifikátor učiaci sa s učiteľom. Na druhej strane, adaptácia bez učiteľa vykazovala podľa očakávaní o niečo horšie výsledky.

Kľúčové slová: EEG, rozhranie mozog–počítač (BCI), binárny klasifikátor

Abstract

In our thesis we used the brain–computer interface to control the ball on the screen in 1D using BCI2000 application. Evaluation of the brain electrical activity (EEG) was based on motor imagery reflected in the spectral density changes in the so-called μ-rhythm measured over the motor cortex. Based on identification of the largest differences in μ-rhythms we extracted features from the measured EEG signal, which served as inputs to the classifier. We focused on methods of feature translation (classification) and replaced the original *supervised* classifier from BCI2000 with the linear discriminant analysis classifier (LDA) which can also be adapted in *unsupervised* ways. Using repeated measurements on four subjects, we evaluated the performance of different methods. We received better results with our supervised LDA classifier compared to the original BCI2000 method. On the other hand, in line with expectations, we observed somewhat worse results with unsupervised LDA classifiers.

Keywords: EEG, brain–computer interface, binary classifier

Predhovor

Vznik tejto témy a následne diplomovej práce bol spojený s iniciatívou doc. Igora Farkaša, ktorý vďaka grantu z Humboldtovej nadácie dokázal zabezpečiť EEG zosilňovač g.tec. S tým bolo spojené rozhodnutie rozbehnúť na Fakulte fyziky, matematiky a informatiky výskum systémov využívajúcich rozhranie mozog–počítač (brain–computer interface, BCI). Táto práca má byť základným kameňom, na ktorom budú v budúcnosti stavať nádejní diplomanti.

Naším cieľom bolo zoznámiť sa s problematikou EEG, merania signálu a jeho spracovania pomocou rôznych aplikácií. Počas našej práce sme objavili aplikáciu BCI2000, v ktorej sme sa rozhodli implementovať náš experiment. Našou úlohou bolo uskutočniť jeden zo základných BCI experimentov a to ovládanie kurzora pomocou mysle. Vybrali sme si metódu založenú na zmenách v tzv. μ-rytme. Subjekt ovládal kurzor pomocou predstavovaného motorického pohybu alebo stavu relaxácie.

EEG signál však nie je stacionárny a BCI systém sa počas experimentu musí byť schopný prispôsobovať sa zmenám v mozgovej aktivite subjektu. Vo všeobecnosti pri BCI systémoch nevieme, čo chce subjekt vykonať a preto sa systém musí byť schopný adaptovať sa aj bez tejto informácie. Preto sme sa rozhodli nahradiť pôvodný klasifikátor z aplikácie BCI2000, ktorý pri adaptácii využíval práve takúto informáciu. Vybrali sme si metôdu lineárna diskriminačná analýza a vyskúšali sme tri rôzne typy adaptácie takéhoto klasifikátora. Výsledky našej práce obsahujú porovnanie piatich rôznych klasifikátorov vyskúšaných na množine subjektov, ich analýzu a vizualizáciu.

Obsah

Úvod

Snaha človeka ovládať počítače alebo iné stroje výlučne pomocou myšlienok sa už roky objavuje v sci-fi filmoch alebo knihách. Vďaka rýchlemu vývoju vedy a techniky sa táto fikcia pomaly stáva skutočnosťou. V ostatnej dobe sa stretávame s rozvojom technológie nazvanej rozhranie mozog–počítač (BCI). Jej hlavnou myšlienkou je nejakým spôsobom získavať signály z mozgu, algoritmicky a pomocou výpočtovej techniky ich spracovávať a prekladať na príkazy, ktoré potom ovládajú nejaký „stroj".

Najväčšie využitie tejto techniky je v medicíne, kde kvôli ochoreniam ľudia prichádzajú o možnosť komunikovať so svojim prostredím alebo vykonávať určité akcie. Boli vyvinuté umelé protézy rúk alebo invalidné vozíky reagujúce na mozgové signály pacienta. Iným príkladom použitia sú technologickí nadšenci, ktorých inšpiruje už len samotná predstava ovládať niečo iba pomocou vlastných myšlienok, a oni vytvárajú počítačové hry, alebo dokonca autá, ktoré sa dajú takto ovládať (Devlaminck et al., 2009).

Cieľom našej práce bolo pomocou aplikácie BCI2000 vyskúšať jednoduchú BCI úlohu, akou je ovládanie kurzora (u nás reprezentovaný guličkou) v 1D priestore. Implementácia experimentu vyžadovala podrobné naštudovanie funkcionality aplikácie BCI2000 a jej jednotlivých častí. Preto bol jedným z cieľov jej podrobný a komplexný opis, ktorého úlohou je uľahčiť prácu osobám, ktoré sa ju rozhodnú využívať. Neskôr sme naimplementovali lineárny binárny klasifikátor, skúšali ho adaptovať rôznymi spôsobmi a vyhodnocovali sme úspešnosť rozličných prístupov pri experimentoch so spätnou väzbou.

Na úvod predstavíme základné meracie metódy mozgu, nami používané aplikácie na prácu s EEG signálom, pojem rozhranie mozog–počítač, jeho typy a trendy vo svete. Druhá kapitola je celá venovaná podrobnému popisu funkcionality aplikácie BCI2000. V tretej kapitole opíšeme metódy používané pri našej práci. V poslednej kapitole popíšeme všetky naše experimenty a vyhodnotíme získané výsledky.

Kapitola 1

Východiská

1.1 Meracie metódy

1.1.1 Signály z mozgu

Vedcov vždy zaujímalo, ako funguje ľudské telo. Často je ich oblasťou záujmu zistiť, ako pracuje mozog, keďže je ústredným orgánom nervovej sústavy človeka a v podstate akési „riadiace centrum“. Mozog je dôležitý aj pre nás z hľadiska témy, takže sa budeme zaoberať práve ním. S pokrokom vedy a techniky získali vedci kvalitnejšie a presnejšie nástroje na jeho spoznávanie a sledovanie procesov v ňom prebiehajúcich. Tieto poznatky využívajú napríklad pri lepšej diagnostike chorôb a nasledovne ich úspešnejšom liečení. Jeden z procesov, ktoré môžeme v mozgu sledovať, je aktivácia jeho jednotlivých častí, čo je pre našu tému opäť najdôležitejšie. Takúto aktiváciu vieme merať viacerými metódami. Podľa spôsobu merania ich vieme rozdeliť na dve hlavné skupiny:

- invazívne
- neinvazívne

Medzi najznámejšie invazívne metódy merania mozgovej aktivity patrí ECoG (electrocorticography). Metóda je podobne ako EEG (elektroencefalograf, pozri časť 1.1.2) založená na meraní elektrických signálov produkovaných mozgom. V porovnaní s EEG sú však elektródy merajúce aktivitu neurónov prikladané subjektu priamo na povrch mozgu. Táto metóda sa väčšinou používa pri ťažkých prípadoch epilepsie, kde je potrebné presne lokalizovať časť mozgu, ktorá spôsobuje silné záchvaty, aby sa nájdené miesto dalo operatívne ošetriť. Taktiež sa táto metóda používa pri študovaní kognitívnych procesov u zvierat. Výhodou metódy je podstatne väčšia presnosť ako pri

EEG, avšak na úkor narušenia lebky.

Ak meriame mozgové signály neinvazívne, tak ich získavame zvonku a do mozgu vôbec nezasahujeme. Väčšinou to prebieha tak, že sa človeku priloží meracie zariadenie na hlavu, ktoré následne meria daný typ signálov. Najpoužívanejšie neinvazívne metódy na získavanie informácií o aktivite mozgu sú:

- NIRS (near-infrared spectroscopy)
- MEG (magnetoencephalgraphy)
- EEG (electroencephalography)

NIRS

NIRS (near-infrared spectroscopy) je technika zisťujúca aktivitu mozgu pomocou sledovania koncentrácie okysličenia hemoglobínu v krvi. Subjekt má na hlave pripevnené meracie zariadenie, ktoré vysiela optické lúče blízke infračerveným (650-900nm) (Chaudhary et al., 2011). Lúč je po prechode mozgom zachytávaný citlivými detektormi a vlastnosti ako pohlcovanie a rozptyl takéhoto lúča určujú množstvo okysličeného (oxy-hemoglobin HbO) a neokysličeného (deoxy-hemoglobin HbR) hemoglobínu. A keďže aktivácia mozgu je priamo spojená so spotrebou kyslíka, tak koncentrácia takéhoto hemoglobínu určuje práve aktivitu mozgu (Vernieri et al., 2004).

MEG

MEG (Magnetoencephalography) je technika na meranie mozgovej aktivity, ktorou sa merajú magnetické polia generované výbojmi neurónov. Meranie prebieha zvonku hlavy pomocou veľmi citlivého prístroja, najznámejší má názov SQUID (superconducting quantum interference device). Takýto prístroj musí byť umiestnený v špeciálnej miestnosti, ktorá je tienená, aby na výsledky merania nevplývali externé vplyvy, ako napríklad magnetické pole Zeme, šum generovaný elektrickými prístrojmi, rádiové signály, alebo nízkofrekvenčné megnetické polia vytvárané pohybujúcimi sa objektmi, ako sú výťahy, autá a vlaky. Príklad takéhoto prístroja je možné vidieť na obr. 1.1.

Výhody MEG:

- meranie prebieha v prirodzenej polohe, v sede a to dovoľuje robiť kognitívne experimenty, ktoré sa viac približujú skutočným životným situáciám, na rozdiel od metód ako je magnetická rezonancia (fMRI), kde je subjekt meraný v ľahu,

Obr. 1.1: SQUID prístroj na meranie MEG signálov.

- meracie zariadenie nevydáva žiadne zvuky, a preto sa táto metóda dá využiť na študovanie reagovania mozgu na zvukové stimiuly,
- má veľké časové (v milisekundách) a taktiež priestorové rozlíšenie,
- signály môžu byť merané z povrchu celej hlavy,
- príprava na meranie je krátka, narozdiel od EEG, kde treba popripájať všetky elektródy vodivým gélom na hlavu.

Hlavným negatívom tejto metódy je potreba tieneného meracieho prostredia. Taktiež nedokáže poskytnúť informácie o štruktúre mozgu. Preto sa často namerané dáta kombinujú s dátami nameranými pomocou magnetickej rezonancie (fMRI). V praxi sa MEG okrem štúdia mozgu používa na detekovanie a lokalizáciu miest, ktoré pacientom s epilepsiou spôsobujú záchvaty. Metóda je užitočná pri hľadaní dôležitých oblastí v prípade odstraňovania mozgových nádorov.

1.1.2 EEG

My sme pri našej práci využívali pravdepodobne najdostupnejšiu metódu - EEG (elektroencefalograf). Je to štandardná neinvazívna metóda merania aktivity mozgu pomocou zaznamenávania elektrických signálov produkovaných neurónmi. Takéto signály sa získavajú pomocou špeciálnych elektród umiestnených na povrchu hlavy. Každá

elektróda meria napätie na mieste, kde je umiestnená. Údaje z elektród spracúva zosilňovač, ktorý signály spracuje, zosilní a cez počítač ich vizualizuje a ukladá. Aby zosilňovač vedel určiť veľkosť signálu, tak z nej nameranú hodnotu porovnáva s inou. Väčšinou sa hodnoty všetkých elektród porovnávajú s údajmi tej istej špeciálnej elektródy, ktorá sa nazýva referenčná (označenie Ref). Na zlepšenie kvality signálu zosilňovač potrebuje ešte jednu elektródu - uzemnenie (označenie Gnd).

História merania EEG siaha až do roku 1924, keď sa nemeckému vedcovi Hansovi Bergerovi podarilo zaznamenať prvý EEG signál. Medzitým doba pokročila a dnes sa EEG využíva na rôzne účely:

- diagnostikovanie epilepsie,
- prognózu pri pacientoch v kóme,
- ako dodatkový test mozgovej zástavy,
- monitorovanie hĺbky anestézie.

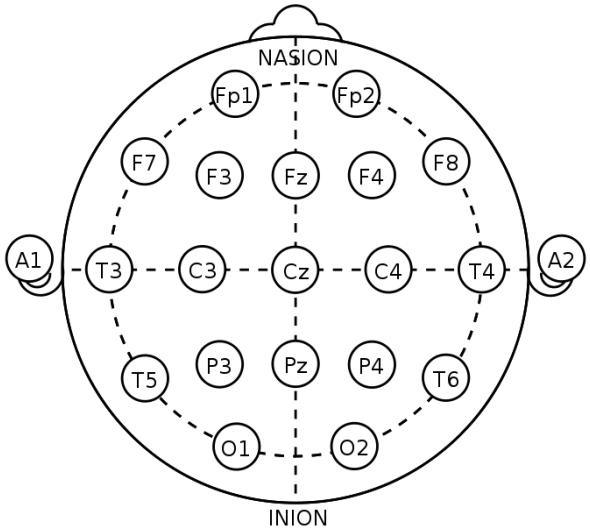
Systém 10–20

Elektródy sa na hlavu umiestňujú podľa toho, čo chceme merať. Existujú rôzne štandardy na ich rozmiestnenie. Sú dôležité kvôli tomu, aby rôzne tímy mohli vykonávať rovnaké experimenty a potom výsledky navzájom porovnávať. Najčastejšie používaným štandardom je systém 10–20 (obr. 1.2). Ide o rozmiestnenie elektród proporcionálne podľa veľkosti hlavy meraného subjektu.

Na obr. 1.2 je vidieť, že pozície, kde umiestňujeme elektródy, môžeme rozdeliť do piatich základných skupín:

- frontálne (F) – nachádzajú sa v prednej časti hlavy,
- centrálne (C) – v strednej časti hlavy,
- occipitálne (O) – v zadnej časti hlavy,
- parietálne (P) – medzi centrálnymi a ocipitálnymi elektródami,
- temporálne (T) – od vrchu hlavy smerom k ušiam.

Pozície začínajúce písmenom A sú na ušiach a tam sa väčšinou pri meraní umiestňuje referenčná elektróda. Po prvom písmene nasleduje buď číslo, alebo písmeno *z* (okrem



Obr. 1.2: Rozloženie elektród podľa systému 10-20.

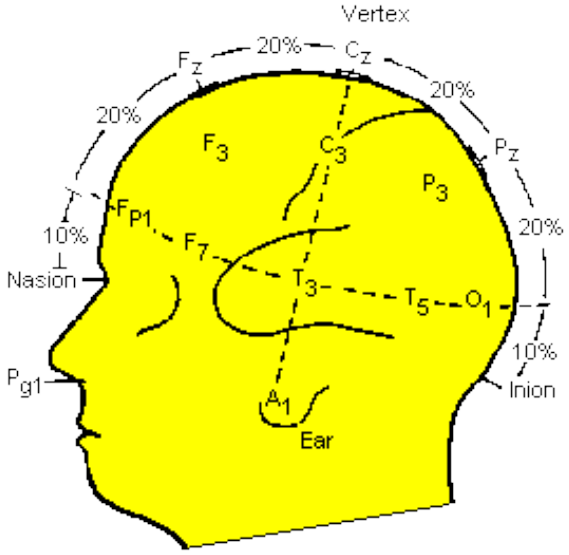
Fp1 a Fp2). Párne čísla nájdeme na pravej hemisfére a nepárne na ľavej. Písmeno *z* označuje stred hlavy pre oblasti F, C a P.

Umiestňovanie elektród týmto systémom sa začína tak, že sa subjektu lokalizujú dva dôležité body: nasion (jamka kde končí nos a začína čelo) a inion (bod na zadnej časti hlavy čo najbližšie ku krku, kde sa končí lebka a začína sa už krk). Zmeria sa vzdialenosť medzi týmito bodmi. Systém bol pomenovaný 10-20 preto, lebo sa elektródy umiestňujú nasledovne: vo vzdialenosti 10% celej dĺžky od bodu nasion sa začína polkruh elektród Fp1, F7, T3, T5 a O1 (symetricky párne čísla), potom o 20% je umiestnená Fz o 20% Cz, o ďalších 20% Pz, potom o 20% končí spomínaný polkruh a o 10% dĺžky je bod inion. V podobnom pomere sú umiestnené nasledovne elektródy T3, C3, Cz, C4 a T4 kolmo na túto priamku (medzi bodmi nasion a inion) a elektródy na už spomínanom polkruhu. Obr. 1.3 názorne ukazuje takéto rozloženie.

Väčšinou sa pri experimentoch kvôli zrýchleniu prípravy subjektu používajú čiapky s pozíciami podľa systému 10-20, na ktoré sa potom umiestňujú elektródy subjektu.

Použité vybavenie

Počas našich experimentov sme využívali merací systém od rakúskej firmy g.tec. Základom bol zosilňovač g.USBamp 3.0 (obr. 1.4). Daný zosilňovač umožňoval meranie 16 kanálov. Pomocou neho sa dalo nahrávať viac druhov signálov ako EEG, ECoG, ECG, EMG, EOG... K počítaču sa pripájal jednoducho pomocou rozhrania USB. Jeho veľkou výhodou bolo, že ak by sme vlastnili viac takýchto zariadení, mohli by sme ich jednoducho pospájať a tým by vzniklo meracie zariadenie s 32, 48 alebo 64 kanálmi.



Obr. 1.3: Rozloženie elektród podľa systému 10-20 zboku.

My sme ho využívali iba na meranie EEG signálov.

Obr. 1.4: Zosilňovač g.USBamp od firmy g.tec.

Do prednej časti prístroja sme zapájali elektródy. Disponovali sme dvoma druhmi elektród:

- pasívne - u nás produkt g.LADYbirdPASSIVE. Bolo to malé kruhové zariadenie (priemer do 2cm), z vnútornej strany obsahujúce prstenec zo striebra alebo chloridu strieborného, pomocou ktorého sa signály prijímali. Aby bolo možné vykonať meranie, musela byť najprv koža subjektu nachádzajúca sa pod elektródou upravená špeciálnym abrazívnym gélom určeným na odstránenie odumretých buniek kože a následne spojená s hlavou vodivým gélom. Po namontovaní pasívnych

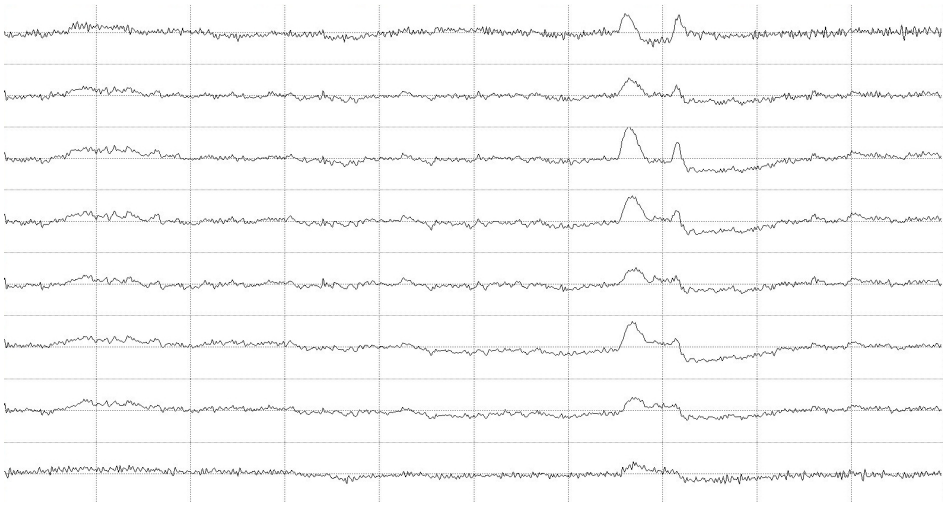
elektród bolo treba pomocou priloženého softvéru skontrolovať impedanciu každej z nich. Aby bol signál presný, tak bolo treba zabezpečiť ich impedanciu menšiu ako 10Ω,

- aktívne - používali sme produkt g.LADYbird. Veľkosťou boli podobné pasívnym, ale na rozdiel od nich priamo obsahovali predzosilňovač, čo spôsobilo, že boli podstatne drahšie, ale produkovali menší šum pri zlej vodivosti medzi elektródou a hlavou, alebo pri pohyboch káblami. Ich montáž na hlavu subjektu bola kratšia, keďže sa používal iný druh gélu a nebolo potrebné pripravovať kožu, ani kontrolovať impedanciu. Kontrola správnosti sa vykonávala iba vizuálne pohľadom na meraný signál.

Elektródy sme pripevňovali na špeciálne čiapky. Mali sme čiapky troch veľkostí, ktoré sme obmieňali podľa veľkosti hlavy meraného subjektu. Pasívne elektródy sa pripájali priamo do zosilňovača, aktívne cez jeden medzičlánok, ktorý pre nich zabezpečoval ovládače a elektrické napätie.

1.1.3 Mozgové vlny

Meraním EEG vzniká záznam - elektroencefalogram. Je to časový priebeh zmien elektrického signálu vytvoreného aktivitou mozgu. Na obr. 1.5 je zobrazený 10 sekundový úsek EEG signálu meraný 8 elektródami (kanálmi).

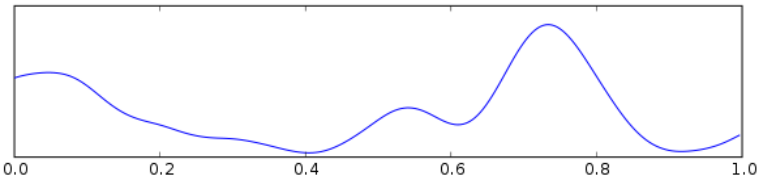


Obr. 1.5: 10 sekundový záznam EEG signálu.

V takomto zázname môžeme identifikovať viacero druhov vĺn. Niektoré sa dajú namerať pri oddychu, niektoré pri pohybe. Ich výskyt tiež súvisí s pozíciou, na ktorej

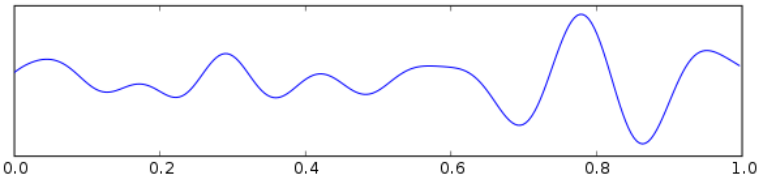
signál meriame. Podľa frekvenčného rozsahu a polohy rozdeľujeme vlny na šesť druhov:

- Delta vlny (do 4Hz) - majú najväčšiu amplitúdu a sú najpomalšie. Najčastejšie ich môžeme vidieť u dospelých počas najhlbších fáz spánku a to nad frontálnou časťou hlavy.



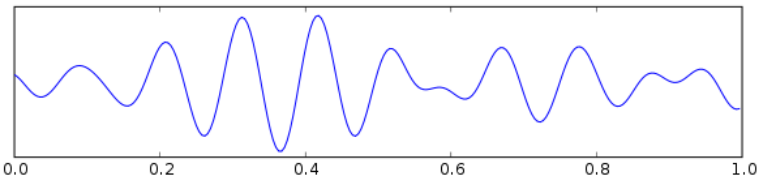
Obr. 1.6: Delta vlny.

- Theta vlny (4–7Hz) - väčšinou sa vyskytujú u malých detí, u starších pri ospanlivosti alebo počas meditácie. Nadmerný výskyt takýchto vĺn signalizuje neobvyklú aktivitu mozgu (choroba). Na druhej strane sa toto frekvenčné pásmo spája s relaxáciou, meditáciou a kreatívnymi stavmi.



Obr. 1.7: Theta vlny.

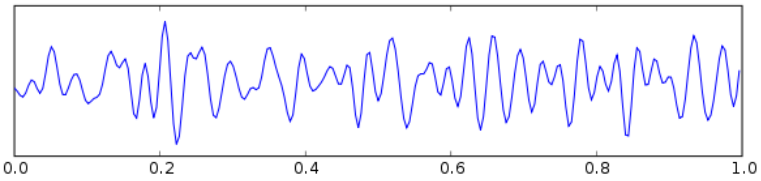
- Alfa vlny (8–12Hz) - sú to prvé namerané vlny (preto ten názov). Viditeľné sú nad obomi hemisférami (vyššia amplitúda nad dominantnou). Prejavujú sa keď zavrieme oči, alebo plytko relaxujeme a sú utlmené, keď oči otvoríme, alebo začneme namáhať mozog.



Obr. 1.8: Alfa vlny.

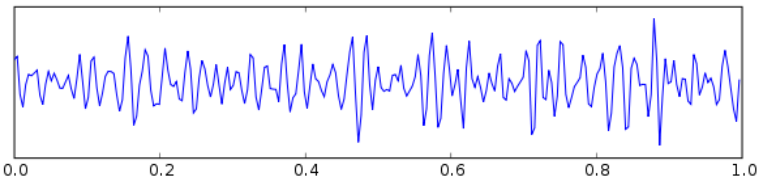
- Beta vlny (12–30Hz) - najčastejšie sú viditeľné symetricky vo frontálnej časti hlavy. Sú spojené s pohybom. Je to asi najbežnejšia mozgová aktivita, ktorú

môžeme pozorovať počas bdelého stavu človeka - najmä pri otvorených očiach, pohybe, prípadne pri koncentrácii.



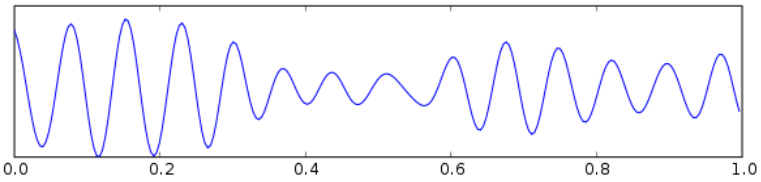
Obr. 1.9: Beta vlny.

- Gama vlny (30–100Hz) - reprezentujú spoluprácu odlišných populácií neurónov do sietí za účelom vykonania určitej kognitívnej alebo motorickej funcie. Taktiež sú viditeľné počas rozpoznávania objektov, zvukov, alebo pocitov.



Obr. 1.10: Gama vlny.

- μ-rytmus (8–13Hz a 18–26Hz) - čiastočne prekrýva ostatné typy vĺn. Je typický pre stav motorického pokoja. Vtedy je synchronizovaný medzi hemisférami. K jeho desynchronizácii dochádza počas vykonávania pohybu, pri jeho pozorovaní a dokonca aj pri jeho predstave.



Obr. 1.11: μ-rytmus.

1.2 Aplikácie používané pri meraní a analýze dát

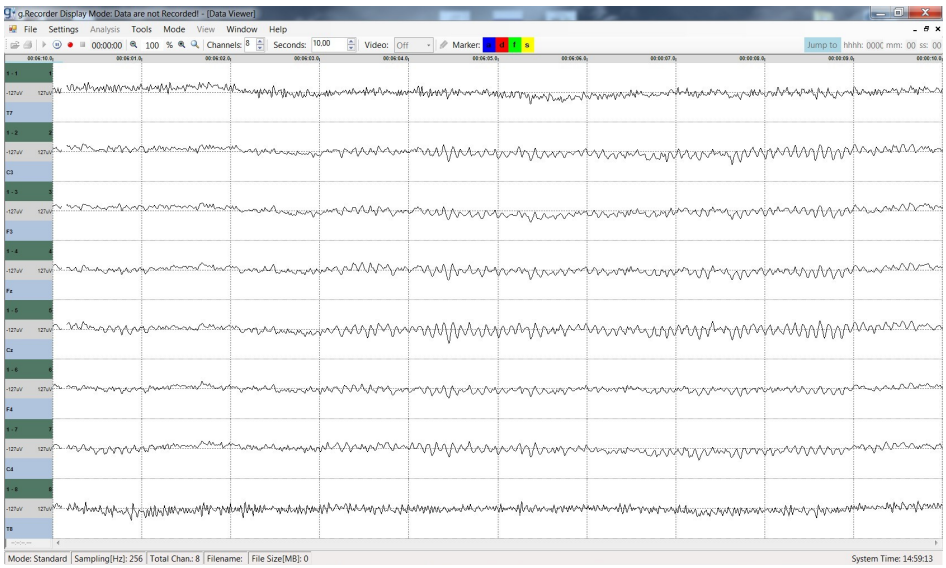
Počas práce sme používali hlavne nasledovné aplikácie:

- g.Recorder,

- Matlab,
- BCI2000.

1.2.1 g.Recorder

Aplikáciu g.Recorder od rakúskej firmy g.tec sme dostali spolu so zosilňovačom. Bol to prvý nástroj na získavanie a zaznamenávanie EEG signálu, ktorý sme používali. Keďže sme najprv používali pasívne elektródy, tak práca s ňou začínala kontrolou impedancie jednotlivých kanálov. Užívateľ si najprv vybral, ktoré elektródy majú byť kontrolované a potom sa postupne meral odpor jednej elektródy za druhou. Odpor jednotlivých elektród bol vyjadrený štyrmi farbami. Modrá farba zmamenala veľký odpor, červená mierne zlepšenie a pri žltej, ktorá reprezentovala odpor menší ako 10kΩ sa už mohlo merať. Posledná farba, zelená, predstavovala odpor menší ako 5kΩ, čo boli najideálnejšie podmienky. Počas našich meraní sme sa snažili pracovať s pripojením elektród dovtedy, kým aplikácia nehlásila zelenú farbu, v nutných prípadoch sme sa zmierili aj so žltou. Okrem kontroly impedancie obsahovala aplikácia aj automatickú kalibráciu zosilňovača.



Obr. 1.12: Základná obrazovka aplikácie g.Recorder.

Pred spustením získavania EEG dát bolo treba aplikáciu správne nastaviť. Náš zosilňovač ponúkal meranie šesťnástich kanálov, väčšinou sa ich však používalo menej. Takže prvým krokom bolo označenie, ktoré kanály sa budú merať. Na jednotlivé kanály sa potom aplikovaním pásmového filtra nastavil rozsah merania (napr. od 0.1–30Hz)

a prípadne použitie tzv. notch filtra, ktorý filtruje frekvenciu, ktrou sa prenáša elektrický prúd v sieti. Jeho hodnota závisí od krajiny, kde sa meria, u nás to je 50Hz, v Amerike 60Hz. Táto frekvencia môže spôsobiť šum v meraných dátach a tým negatívne ovplyvniť výsledky experimentov. Každý kanál sa pre prehľadnosť dal premenovať.

Hlavnou funkciou aplikácie g.Recorder je získavať, zobrazovať a zaznamenávať dáta zo zosilňovača. Základnú obrazovku môžete vidieť na obr. 1.12. Najväčšiu časť obrazovky tvorí zobrazenie meraného signálu. V prípade, že je meraných priveľa kanálov, tak sa dá nastaviť, ktoré majú byť viditeľné na obrazovke a ktoré nie. Taktiež sa dá nastaviť rozsah, v ktorých signál zobrazujeme, my sme pracovali s rozsahom od -127 do 127μV. Okrem signálu sa na obrazovke nachádzajú ovládacie prvky. Pomocou nich môžeme spustiť zobrazovanie signálu, prípadne jeho nahrávanie. Aplikácia primárne ukladá dáta do formátu *.hdf5, ktorý dokáže Matlab bez problémov prečítať. Uložené dáta sa dajú v aplikácii spätne prehrať. Pri meraní môžeme do signálu pomocou dopredu nastavených klávesových skratiek vkladať značky, ktoré nám môžu označovať dôležité udalosti, napr. kedy začal subjekt niečo vykonávať a podobne. Takáto informácia sa dá neskôr spracovať a pomocou nej môžeme z pôvodného signálu „vystrihnúť“ časť, ktorá je niečim špecifická, napr. *subjekt vtedy hýbal pravou rukou.*

1.2.2 Matlab

Matlab (skratka z *matrix laboratory*) je jedno z najpoužívanejších prostredí určených na matematické výpočty a zároveň skriptovací programovací jazyk. Jeho vývoj začal v sedemdesiatych rokoch ako balík počítajúci systémy lineárych rovníc a vlastné hodnoty. Teraz je to produkt plný funkcionality ponúkajúci moderné knižnice, ktoré sa dajú aplikovať na rôzne oblasti ako sú finančná matematika, neurónové siete, alebo teória riadenia. Má vstavaný interpretovaný jazyk podobný Fortranu90 a vďaka flexibilnému indexovaniu matíc je vhodný na implementáciu maticových problémov. Svoju popularitu získal vďaka jednoduchému a intuitívnemu používaniu. Bežne sa používa pri výučbe napríklad aj u nás na FMFI. Taktiež sa v ňom dobre pracuje s grafmi a preto sa často používa ako nástroj na analýzu dát.

Kvôli tejto funkcionalite sme ho používali aj my. Dáta namerané v aplikácii g.Recorder (pozri časť 1.2.1) sme načítali do Matlabu a následne ich analyzovali. Počas celej práce sme vytvorili množstvo skriptov napísaných v Matlabe, ktoré nám uľahčovali našu prácu. Mali sme skripty určené napr. na konverziu dát z formátu *hdf5 do *mat

(základný formát pre Matlab, v ktorom sa ukadajú matice), na vykreslenie nameraných dát, alebo na počítanie spektrálnej analýzy získaného signálu.

Výpočet spektrálnej hustoty

Tento skript bol jeden z najpoužívanejších hlavne v začiatkoch našej práce. Vtedy sme všetky merania robili pomocou aplikácie g.Recorder a následne ich spracovávali týmto skriptom. Jeho tvorba bola inšpirovaná diplomovou prácou Tomáša Debnára (Debnár, 2010). Výsledky použitia tohto skriptu sa nachádzajú v časti 4.1. Pri jeho písaní sme predpokladali, že sa bude vždy merať dopredu daný počet stavov a všetky budú mať rovnakú dĺžku.

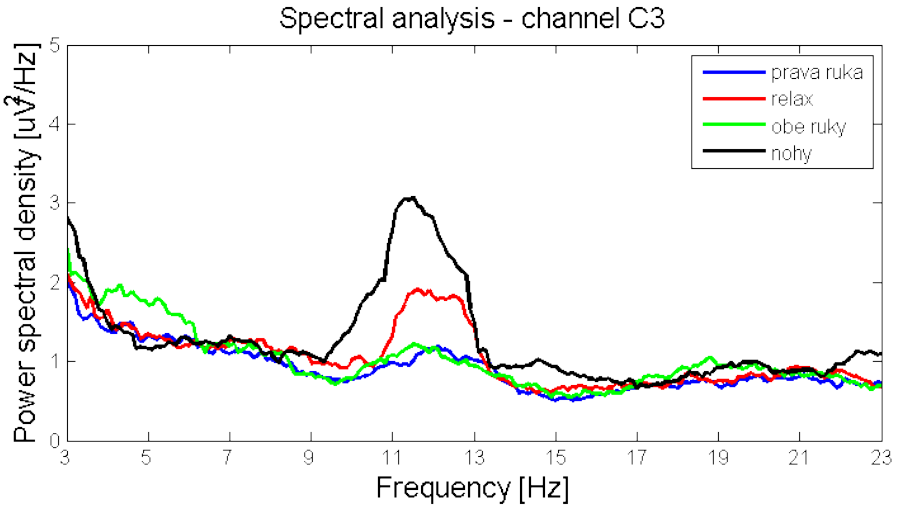
Pri spektrálnej analýze bolo východiskom tvrdenie, že každý EEG signál sa dá vyjadriť ako súčet pravidelných sínusoíd rôznych frekvencií a amplitúd. Nás zaujímalo, že ako často sa v signáli nachádzajú sínusoidy určitej frekvencie. Preto má výsledok spektrálnej analýzy na jednej osi jednotlivé frekvencie a na druhej „silu“ danej frekvencie. Čím viac sínusoíd danej frekvencie sa v signáli nachádza, tým väčšiu má daná frekvencia silu (Cimrová, 2011).

Skript sme spúšťali v Matlabe nasledovným volaním:
`spectralAnalysis(fileName, paramName)`. Mal dva vstupné parametre. Ako `fileName` bolo potrebné zadať názov vstupného súboru bez prípony (vo formáte *hdf5), ktorý obsahoval dáta, z ktorých sme chceli počítať spektrálnu analýzu a `paramName` udával cesta k súboru s parametrami (textový súbor v určenom formáte). Ten v každom riadku obsahoval jeden parameter zapísaný pomocou troch častí. Ako prvá začínala hodnota, po jednej medzere nasledoval názov parametra (napr. useMarkers) a nakoniec po znaku % pokračoval komentár. Parametre mohli obsahovať číselné (napr. cutBeg, ktorý vyjadroval, koľko sekúnd záznamu zo začiatku súboru bolo treba odrezať), alebo logické (napr. count vyjadroval, či bolo treba skutočne počítať) hodnoty. Posledné dva parametre súboru boli legenda a názvy meraných elektród. Medzi najdôležitejšie parametre patrili numCond, ktorý hovoril, koľko stavov sme merali, a condLen hovoriaci, koľko každý stav trval.

Skript po konverzii načítal dáta zo vstupného súboru aj parametre a signál predspracoval. Ako sme spomínali v časti 1.2.1, počas merania sme mohli do signálu vkladať značky, ktoré označovali dôležité časti. Ak bol parameter useMarkers nastavený na TRUE, tak sa zo vstupných dát „vystrihli“ časti zodpovedajúce jednotlivým stavom. Stav začínal značkou a pokračoval počtom sekúnd nastaveným parametrom numCond. V prípade nepoužitia značiek sa signál „rozrezal“ na jednotlivé stavy iba podľa ich

dĺžky.

Takto pripravené dáta rozdelené podľa stavov sa pošlali do funkcie countSpectrum, ktorá vypočítala samotnú spektrálnu analýzu. Vrátené dáta sa zobrazili do grafov. Vytvoril sa samostatný graf pre každý meraný kanál. V každom grafe sa potom nachádzala jedna krivka pre každý zo stavov (pozri obr. 1.13).



Obr. 1.13: Výsledok spektrálnej analýzy na kanáli C3 pre štyri rôzne stavy: predstavovaný pohyb pravou rukou, relax, predstavovaný pohyb oboma rukami a predstavovaný pohyb nohami.

1.2.3 BCI2000

Aplikáciu BCI2000 sme používali až od polovice našej práce na prácu s rozhraním mozog–počítač (popísaný v časti 1.3). Podrobný popis jej významu, funkčnosti a použitia sa nachádza v kapitole 2.

1.3 Rozhranie mozog–počítač

1.3.1 Čo to je

Mnoho ľudí trpí neurologickými poruchami, ktoré narúšajú prirodzený tok nervovej aktivácie z mozgu cez miechu na určené miesto, napr. prst na ruke. Takéto poruchy mohlo spôsobiť zranenie miechy, mozgová príhoda, alebo iné ochorenia oblastí, ktoré prenášajú samotnú informáciu, napr. ochorenie samotných svalov alebo

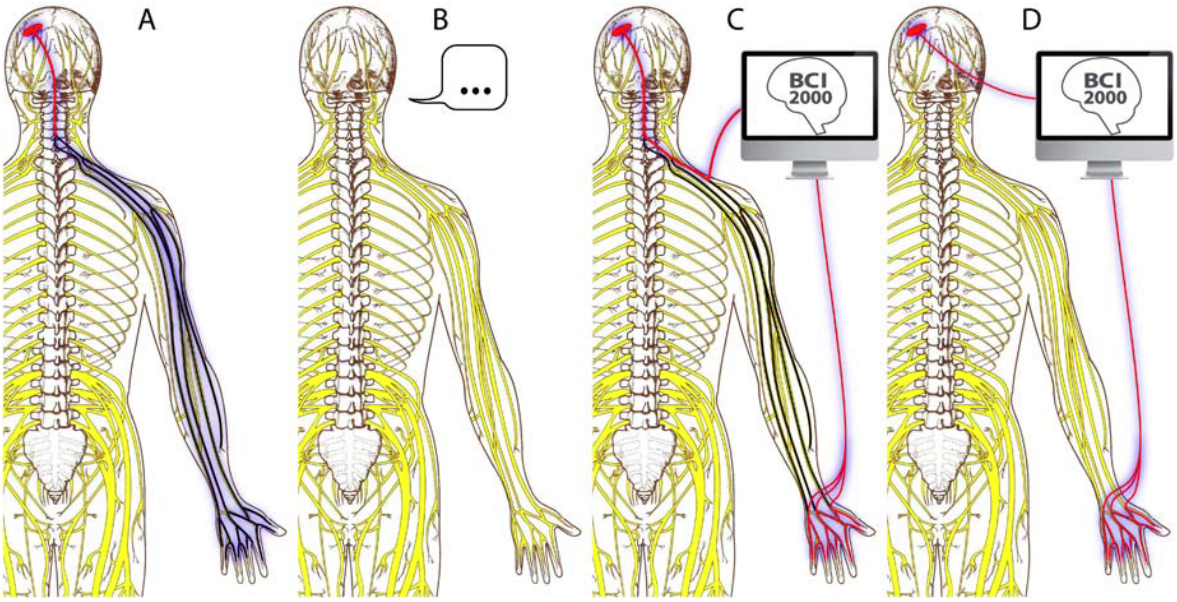
nervových dráh. Takíto pacienti stratili možnosť používať svaly a tak prišli o možnosť komunikovať alebo interagovať so svojim prostredím pomocou pôvodných *rozhraní*. V takýchto prípadoch môže byť nápomocná veda a technika. Snahou vedcov bolo vytvoriť pre takýchto pacientov nejaké umelé rozhranie, pomocou ktorého by mohli komunikovať alebo interagovať s prostredím tak, ako boli pôvodne zvyknutí.

Existuje viacero spôsobov ako riešiť takýto problém. Prvou možnosťou je ovládať postihnutú časť tela (napr. ruka, do ktorej nejdú informácie z mozgu) takými nervami a svalmi, ktoré ešte stále fungujú. Napríklad pacient môže používať pohyb očí alebo rúk na komunikáciu s prostredím. Iným príkladom môže byť ovládanie robotickej protézy ruky rečou, pozri obr. 1.14B. Takáto náhrada sa dá využiť iba v určitých prípadoch, napriek tomu môže byť pre pacienta veľmi prospešná. V prípadoch, keď vieme presne určiť miesto poškodenia cesty od mozgu k cieľu (napr. ruka), tak môžeme poškodenú cestu nahradiť externou. V prípade poškodenia ruky v oblasti medzi ramenom a lakťom by to vyzeralo tak, že by pacient mal v ramene voperovaný implantát, ktorý by meral aktivitu príslušných (funkčných) nervov, danú informáciu by posielal nejakému výpočtovému zariadeniu, ktoré by informácie spracovávalo a vyhodnocovalo a výstup by posielalo do druhého implantátu v spodnej časti ruky (napr. predlaktie), kde sú nervy opäť funkčné, ktorý by dané nervy stimuloval a tým by sa informácia dostala do cieľa (napr. mohol by hýbať prstom), pozri obr. 1.14C. Poslednou možnosťou je vytvorenie úplne nového externého kanálu priamo z mozgu do cieľa, pozri obr. 1.14D.

Práve touto možnosťou sa budeme zaoberať v tejto práci. Základnou myšlienkou je nejakou metódou získavať signály z mozgu pacienta, tie pomocou počítača spracovávať a prekladať ich do príkazov, pomocou ktorých sa vykonávajú požadované akcie. Keďže sú signály získavané priamo z mozgu a využíva sa na ich spracovanie výpočtová technika, takýto spôsob bol nazvaný rozhranie mozog–počítač (brain–computer interface, BCI).

Rozhranie mozog–počítač je komunikačný systém, pri ktorom sa nevyužívajú svaly a subjekt ho môže použiť na prenos svojho zámeru z mozgu do prostredia. Keďže je BCI systém napojený priamo na mozog subjektu, vzniká tým komunikačný kanál medzi mozgom a počítačom. Ako každý komunikačný systém, BCI má vstupy (získané mozgové signály), výstup (riadiace signály, ktorými sa ovláda nejaké zariadenie, napr. invalidný vozík) a komponent, ktorý prekladá jedno na druhé (počítač). Preto štandardný BCI systém opakuje nasledovné tri kroky:

- získanie signálu z mozgu (signal acquisition),

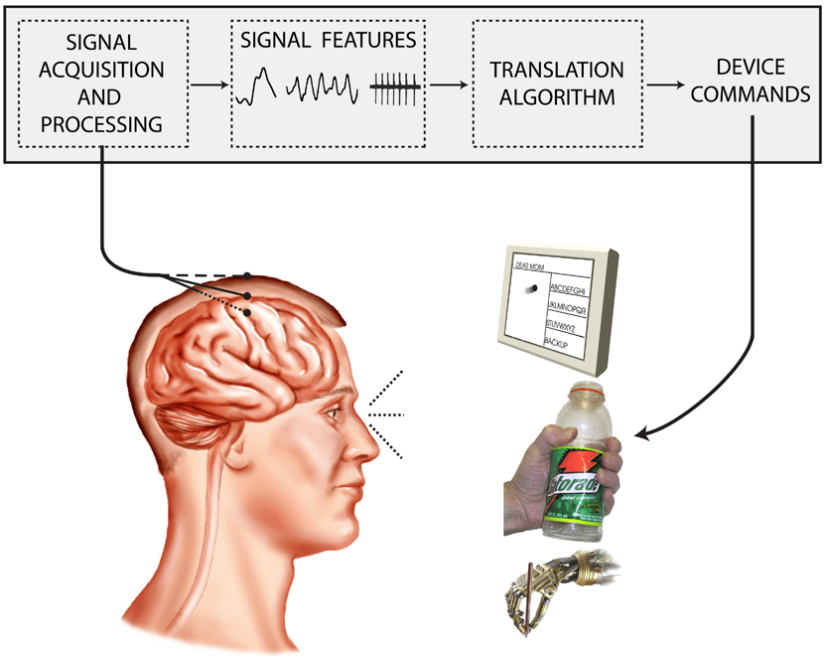


Obr. 1.14: Možnosti pri paralyzovanej ruke. A: Normálny tok informácií medzi mozgom a cieľom - dlaňou je poškodený. B: Ovládanie pomocou iných ešte stále funkčných rozhraní (napr. reč). C: Nahradenie pôvodnej cesty externým výpočtovým zariadením. D: Vytvorenie nového komunikačného kanálu priamo z mozgu do cieľa - rozhranie mozog–počítač (BCI). Obrázok prevzatý zo Schalk and Mellinger (2010).

- extrahovanie príznakov zo signálu (feature extraction),
- preloženie vybraných príznakov do riadiacich signálov (feature translation).

BCI systém je vlastne uzavretý cyklus týchto troch činností. BCI pomocou zosilňovača získa signály z mozgu, vyberie z nich charakteristické príznaky, preloží ich algoritmami do riadiacich signálov a nimi ovláda nejaké koncové zariadenie (napr. invalidný vozík, kurzor na obrazovke...). Subjekt pomocou spätnej väzby (sleduje, čo robí aplikácia) upravuje svoju mozgovú aktivitu a cyklus sa opakuje (obr. 1.15). Ako sme spomínali v časti 1.1.1, možností ako merať signály od subjektu je viac. Kvôli dostupnosti sa najčastejšie využíva EEG.

Pri meraní kvality BCI systému sú najdôležitejšie rýchlosť (ako dlho trvá jeden cyklus od namerania signálov, spracovanie až po vykonanie akcie) a presnosť (ako presne systém interpretuje úmysly subjektu do vykonávaných akcií). BCI systémy používané v dnešnej dobe dokážu v priebehu pár sekúnd dosahovať vysokú presnosť, napr. pri klasifikovaní do dvoch tried až 90% (pri takýchto experimentoch je referenčnou hodnotou 50%, čo zodpovedá náhodnej voľbe triedy).



Obr. 1.15: Základný design a funkčnosť BCI systému. Signály sú získavané z mozgu, následne sa z nich extrahujú príznaky, ktoré su prekladané do ovládacích príkazov, ktoré ovládajú nejakú koncovú aplikáciu. Počas celého procesu subjekt pomocou spätnej väzby reaguje na správanie aplikácie a na základe toho upravuje svoju mozgovú činnosť. Obrázok prevzatý zo Schalk and Mellinger (2010).

1.3.2 Paradigmy

Existuje viac základných druhov BCI (paradigiem), ktoré sa odlišujú tým, ako počítač „zistí“, čo človek chcel spraviť. Typy vieme rozdeliť do dvoch skupín, podľa toho, či snímané signály z mozgu vyvolal priamo subjekt, alebo boli vyvolané nejakými externými podnetmi (Devlaminck et al., 2009). V každej skupine uvedieme niekoľko príkladov, ktoré neskôr podrobne opíšeme.

- signály vznikajú pomocou externých stimulov
  - P300
  - SSEP (Steady-State Evoked Potential)
- signály vyvolané subjektom
  - SCP (slow cortical potentials)
  - ERD (event-related desynchronization)

P300

Táto metóda je založená na reakcii mozgu, keď dostane nečakaný stimul. Subjekt dostáva nejaké stimuly (audiálne, vizuálne...), nazvime ich štandardné stimuly. Napríklad počuje tóny s nízkou frekvenciou. Občas však dostane (s malou frekvenciou napr. 20%) iný stimul - cieľový stimul (vysoký tón). Vedci prišli na to, že asi 300ms po cieľovom stimule sa dá namerať pozitívne napätie nad centrálnou a parietálnou časťou mozgovej kôry.

Ako prvou aplikáciou, ktorá využívala túto metódu bol P300 speller. Je to hláskovacie zariadenie, ktoré pomáha ľuďom, ktorí nemôžu inak komunikovať s okolím. Subjekt vidí na obrazovke znaky (písmená, čísla, medzeru...) usporiadané do riadkov a stĺpcov. Subjekt si zvolí jeden znak. Potom sa postupne zvýrazňujú (menia farbu, veľkosť) jednotlivé riadky a stĺpce (obr. 1.16).

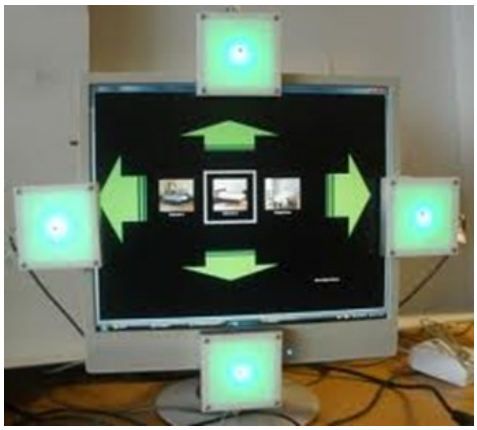
Obr. 1.16: Užívateľské prostredie aplikácie P300 speller. Obrázok prevzatý zo Devlaminck et al. (2009).

Úlohou subjektu je počítať, koľkokrát sa vysvietil jeho znak (buď v stĺpci alebo riadku). Počítač sleduje zvýšenie napätia a keď sa postupne vysvietia všetky stĺpce a riadky, tak určí, ktorý znak si subjekt vybral.

Steady-state evoked potential

Metóda Steady-state evoked potential je založená na tom, že keď človek prijíma stimuly zvonku určitej frekvencie, tak sa aj neuróny v mozgu aktivujú v rovnakej frekvencii. Takto sa zvýši meraný „výkon“ na danej frekvencii. Príkladom využitia je ovládanie aplikácie, kde sú ponúkané možnosti hore, dole, doprava a doľava. Na obra-

zovke počítača je spustená aplikácia a na krajoch obrazovky sú umiestnené zariadenia, ktoré blikajú rôznou frekvenciou (obr. 1.17). Ak subjekt prijíma vizuálne stimuly, tak sa metóda nazýva presnejšie SSVEP (Steady-State Visual Evoked Potential).



Obr. 1.17: Príklad BCI typu SSEP.

Užívateľ ovláda aplikáciu tým, že prijíma jeden z ponúkaných stimulov (napr. zahľadí sa na jeden z blikajúcich objektov). Program zistí, akou frekvenciou je prijímaný stimul a podľa toho vykoná akciu. Táto metóda je obľúbená hlavne kvôli vysokej presnosti a malému času, ktorý je potrebný na učenie subjektu.

Slow cortical potentials

Slow cortical potentials sú pomalé kladné alebo záporné posuny EEG signálu. Trvajú od 0,3 až po niekoľko sekúnd. Záporné posuny reprezentujú mobilizáciu neurónov pred vykonávaním nejakej kognitívnej úlohy. Kladné posuny sa dajú namerať počas vykonávania samotnej úlohy a preto reprezentujú „spotrebu mozgových prostriedkov“. Ľudia sú schopní naučiť sa ovládať tieto posuny pomocou rozsiahleho učenia sa s odozvou a takto napríklad ovládať aplikáciu na písanie slov pomocou hláskovania. Bolo dokázané, že ich ovládanie môže pomôcť pacientom s epileptickými záchvatmi alebo hyperaktivitou utlmovať príznaky ich choroby.

Event-related desynchronization

Metóda *Udalosťou vyvolaná desynchronizácia* je založená na zmenách v μ-rytmu (pozri časť 1.1.3). Pri pohybe človeka (napr. rukou) dochádza k udalosti nazvanej ERD (event-related desynchronization) - pokles napätia nad motorickou časťou mozgovej kôry vo frekvenčnom pásme μ-rytmu (hlavne 8–12Hz). Po dokončení pohybu

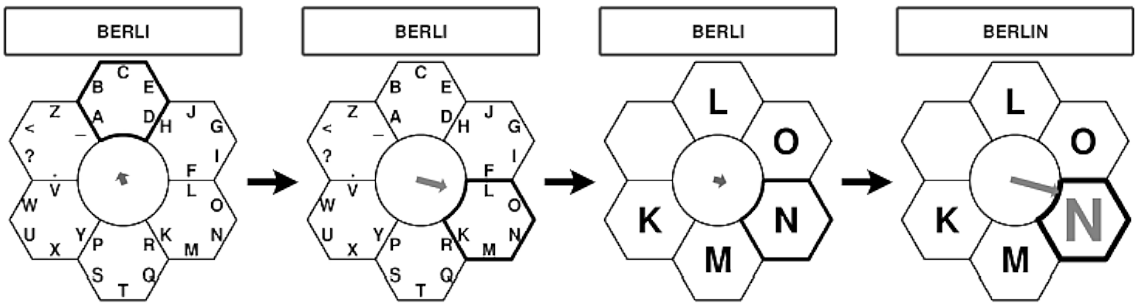
a následnej relaxácii dochádza k ERS (event-related synchronization) - zvýšeniu napätia na pôvodnú hodnotu.

Pozícia na hlave, kde sa dá takáto zmena namerať, súvisí s typom pohybu, pri ktorom chceme zmenu pozorovať. Ak subjekt pohne ľavou rukou, tak by mal byť pokles najvýraznejší nad pravou hemisférou okolo elektródy C4. Symetricky to je s pravou rukou. Ak sledujeme zmenu pri pohybe nohou, tak by sa mala najviac prejaviť medzi strednou elektródou Cz a elektródami C3 alebo C4 podľa toho, či to bola pravá alebo ľavá noha.

Na tomto založené BCI funguje len preto, že zmena μ-rytmu nenastáva iba keď vykonávame pohyb, ale takisto aj keď pohyb pozorujeme (predpokladá sa, že sú za to zodpovedné zrkadliace neuróny, tie majú spojenia na motorickú kôru v ktorej svojou aktivitou vyvolávajú ERD), alebo dokonca keď si ho len predstavujeme (Pineda, 2005). Človek bez nohy nemôže samotný pohyb vykonávať, ale ak ho už niekedy vykonával (nenarodil sa už bez nohy), tak by si ho mal byť schopný predstaviť. A tak môže pomocou BCI ovládať napríklad umelú nohu.

Táto metóda však nie je tak presná ako napríklad P300 a aj doba potrebná na naučenie subjektu je vyššia.

Príkladom implementácie takejto metódy je aplikácia Hex-o-Spell (Blankertz et al., 2006) vyvinutá v Berlíne. Je to tiež aplikácia určená na písanie slov pomocou hláskovania, podobne ako P300 (1.3.2). Naučeným subjektom sa podarilo písať rýchlosťou až 7,6 znaku za minútu. Užívateľ vidí šesť hexagónov v ktorých sú napísané znaky (písmená, čísla, medzera...) a medzi nimi v strede je šípka (obr. 1.18).



Obr. 1.18: Aplikácia Hex-o-Spell. Obrázok prevzatý z Blankertz et al. (2006).

Aplikácia sa ovláda dvoma predstavovanými pohybmi. Ak subjekt chce, aby sa šípka v strede otáčala v smere hodinových ručičiek, tak si predstavuje pohyb ľavej ruky. Ak si želá, aby sa šípka prestala otáčať a aby sa predĺžila (zväčšila) a tým vybrala jeden hexagón, tak si predstaví pohyb pravej nohy. Vtedy sa vyberie jeden hexagón, v ktorom

bola sada znakov. V druhom kroku sa zobrazí opäť šesť hexagónov, ale v každom už bude len jeden znak (z hexagónu vybratého v prvom kroku). Užívateľ si opäť vyberie jeden hexagón a tak napíše znak. Práve túto paradigmu založenú na predstavovanom pohybe sme používali v našej práci.

1.3.3 BCI vo svete

Kľúčovým prvkom robustného a úspešného BCI systému je modul zodpovedný za spracovanie signálu z mozgu, ktorý z neho extrahuje príznaky a následne prekladá do riadiacich signálov. Preto sa tejto časti momentálne venuje najviac pozornosti. Najčastejšie sa hľadajú odpovede na nasledovné otázky:

- **Ktoré techniky spracovania signálu si zaslúžia viac pozornosti?** V súčasných BCI systémoch sa pri získavaní príznakov často používa spektrálna analýza. Táto metóda má však obmedzené možnosti ako zachytiť časové závislosti v signáli. Preto v článku Krusienski et al. (2011) navrhujú preskúmať, či by mohli byť rekurentné neurónové siete vhodnou alternatívou.

- **Prečo niektorí ľudia nie sú schopní ovládať BCI?** Tomuto problému sa venujú viaceré tímy, napr. Vidaurre et al. (2010). Jedným z možných vysvetlení je prechod medzi offline fázou trénovania a online pokusmi so spätnou väzbou. V článku skúšali adaptatívne metódy strojového učenia, aby sa vyhli fáze offline kalibrácie. Vďaka tomu sa im podarilo v priebehu 60 minút natrénovať problematické subjekty tak, aby boli schopné ovládať ich BCI systém.

- **Ako môžeme adaptáciou príznakov a klasifikátora reagovať na nestacionaritu EEG signálu?** Za hlavnú výzvu na to, aby sme mohli BCI systémy používať v reálnom živote, sa považuje robustná a úspešná adaptácia, ktorá nepotrebuje informáciu o zámere subjektu. Pri takejto odaptácii sú najdôležitejšie zmeny kovariácie prvkov, adaptácia príznakov a adaptácia klasifikátora.

Okrem toho vyberáme dve zaujímavé použitia BCI systémov:

Ovládanie auta vo virtuálnom priestore

Vedcom z Číny a Japonska sa podarilo vytvoriť BCI systém založený na ERD metóde, pomocou ktorého dokázali ovládať auto vo virtuálnom 3D prostredí (Zhao et al., 2009). Na ovládanie používali 4 príkazy: predstavovaným pohybom ľavou alebo

pravou rukou vozidlo zatáčalo, predstavovaným pohybom nohou zrýchľovalo a pri relaxácii sa nevykonávala žiadna z predchádajúcich akcií. Merali 5 elektród nad motorickou časťou mozgu. Zaujímavé bolo, že nerozlišovali iba medzi danými štyrmi stavmi, ale sledovali aj dĺžku vykonávaného stavu a tým dokázali plynulo zatáčať, alebo zrýchľovať.

Počas svojich experimentov testovali 4 subjekty a ich úlohou bolo najprv na rovnej ceste kľučkovať pomedzi kužele. Neskôr museli ovládať auto na vyznačenej trati obsahujúcej ostré zákruty a snažiť sa dosiahnuť čo najlepší čas. Úspešné výsledky klasifikácie nad 75% dosiahli 3 zo 4 subjektov.

Projekt Brain Driver

V laboratóriu AutoNOMOS, ktoré je súčasťou Berlínskej univerzity, sa podarilo vytvoriť prototyp autonómneho auta. Išlo o upravenú verziu auta Volkswagen Passat, ktoré používlo množstvo senzorov na automatický pohyb po cestných komunikáciach.

Súčasťou toho veľkého projektu bolo aj využitie BCI systému. ERD metódu použili na rozhodovanie sa, kam má auto na najbližšej križovatke odbočiť. Pracovali s binárnym klasifikátorom v 1D, ktorý neskôr rozšírili do 2D priestoru a použili na priame ovládanie volantu vozidla a jeho akceleráciu. Na meranie signálu použili zatiadenie EPOC od austrálskej firmy Emotive, pomocou ktorého merali 14 kanálov.

1.4 Ciele práce

Cieľom našej práce bolo s využitím meracieho zariadenia EEG implementovať učiaci systém s rozhraním mozog-počítač (BCI), ktorý by bol schopný využiť spätnú väzbu na realizáciu akcií pomocou mysle. Na to sme potrebovali:

- naštudovať si literatúru z oblasti BCI systémov (metódy merania EEG, algoritmy spracovania a vyhodnocovania signálu),
- zoznámiť sa s meracím zariadením EEG,
- získať praktické skúsenosti s meraním,
- otestovať funkčnosť systému na jednoduchšej úlohe (napr. ovládanie kurzora myši na obrazovke).

Kedže sme na imlementáciu BCI systému použili aplikáciu BCI2000, tak ďalším z cieľov bolo preskúmať jej funkcionalitu a zamerať sa na časť, pomocou ktorej sa

dal vytvoriť experiment, kde úlohou subjektu bolo ovládať kurzor na obrazovke, čo vyžadovalo prečítanie dokumentácie a občas čítanie zdrojových kódov. Najdôležitejšie bolo zistiť, ako presne sa spracováva signál a akú funkcionalitu vykonávajú jednotlivé časti aplikácie.

Ako posledný cieľ sme si stanovili implementáciu binárneho klasifikátora, ktorý by sa dal adaptovať metódou *učenie bez učiteľa* a porovnať úspešnosť rôznych metód adaptácie medzi sebou.

Kapitola 2

BCI2000

2.1 Úvod

BCI2000 je aplikácia určená na výskum BCI. V roku 2001 vznikla a stále sa vylepšuje vo Wadsworth Center of the New York State Department of Health in Albany, New York, USA v spolupráci s nemeckou univerzitou v meste Tübingen pod vedením Gerwina Schalka, Ph.D (Schalk and Mellinger, 2010). Jej vznik bol podnietený potrebou robustného systému určeného na výskum a vývoj BCI systémov tak, aby na nich mohli spolupracovať viaceré tímy súčasne. V čase zrodenia myšlienky vytvoriť takýto systém existovali iba úzko špecializované softvéry, ktoré implementovali len jednu konkrétnu BCI paradigmu. Snahou tvorcov BCI2000 bolo vytvoriť všeobecnú platformu, pomocou ktorej sa budú dať vyvýjať rôzne typy BCI systémov (založené na desynchronizácii μ-rytmu, P300...) a bude na nich môcť spolupracovať veľká skupina ľudí.

Aplikácia je pre výskum a výučbu voľne dostupná. V čase písania tejto práce ju používalo viac ako 600 laboratórií po celom svete. Je naprogramovaná v programovacom jazyku C++ a pri kompilácii používa program CMake (Cross Platform Make). Vďaka tomu je možné vytvoriť špecifické Makefiles a tak vytvoriť projekty pre rozličné vývojové prostredia a operačné systémy. Momentálne sú dostupné tieto prostredia (udávané výrobcom):

Podporované operačné systémy

- Windows XP, Windows 2000 (Vista and Windows 7 nie sú odporúčané pre BCI2000 kvôli slabým audiovizuálnym latenciám)
- 64-bitové Windows systémy (s obmedzeniami)

- Macintosh OS X (väčšinou bez problémov, ale zatiaľ nie je oficiálne podporovaný)
- Linux (skompilovateľné, ale vyskytujú sa problémy so sieťou)

Podporované kompilátory

- Borland/CodeGear/Embarcadero C++ Builder 2010 (bez GUI aplikácií); C++ Builder XE2 je nepodporovaný
- Visual Studio 9 (2008) a 10
- MinGW s gcc 4.x

Podporované vývojové prostredia

- Borland/CodeGear/Embarcadero 6, 2007, 2009, 2010
- Visual Studio 9 (2008), Visual Studio 10 (pomocou CMake)
- Code::Blocks - MinGW Makefiles (pomocou CMake)
- Iné vývojové prostredia podporované CMake generátormi (napr. Eclipse CDT) kým používajú vyššie uvedené kompilátory

My sme pri práci používali 32-bitovú verziu Windows 7 a hoci nebol odporúčaný, tak sme s ním nespozorovali žiadne problémy. Program sme upravovali vo Visual Studiu 2008. Najaktuálnejšia verzia BCI2000 je 3.0.5, ktorá vyšla v júli 2012.

Program je možné získať z domácej stránky projektu http://www.bci2000.org. Ak chce niekto BCI2000 stiahnuť, tak je potrebná najprv bezplatná registrácia a potom pomocou prihlasovacích údajov stiahnutie inštalačného balíka. Po inštalácii vznikne v užívateľom určenom priečinku nasledovná súborová štruktúra:

- batch – obsahuje *.bat súbory určené na rýchle spúšťanie predvytvorených experimentov,
- data – tu sa ukladajú dáta vytvorené pri experimentoch,
- doc – HTML nápoveda a článok (Schalk et al., 2004),
- params – parametre používané pri experimentoch (napr. nastavenia zosilňovačov),
- prog – spustiteľné programy používané v BCI2000,

- tools – užitočné pomôcky pri práci s BCI2000 ako prehliadač, nameraných dát alebo nástroj na offline analýzu.

Aplikácia BCI2000 obsahuje všetky potrebné časti na to, aby bolo možné skúšať dva základné experimenty: BCI založené na P300 (písanie textu pomocou hláskovania, vysvetlené v časti 1.3.2) a BCI založené na desynchronizáci μ-rytmu (princíp popísaný v časti 1.3.2). Vybratý experiment sa spúšťa *.bat súborom. V priečinku batch sa však nachádza viac súborov pre jeden experiment (väčšinou štyri). Zvolenie závisí od názvu používaného zosilňovača. Daný súbor spustí všetky potrebné moduly a načíta parametre zo súborov.

Samotná aplikácia obsahuje menu a potom štyri základné tlačidlá:

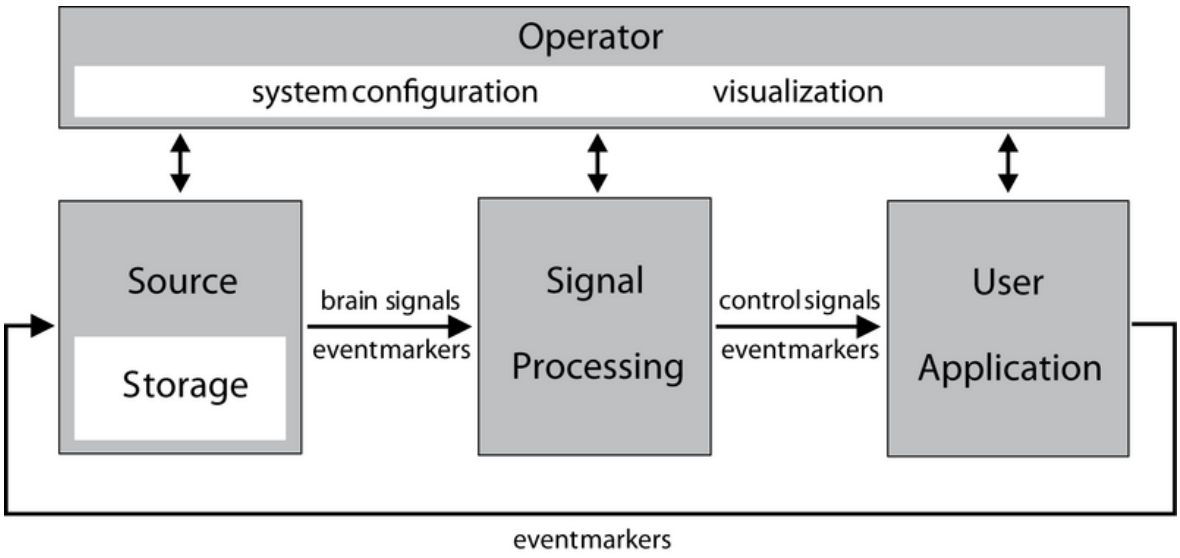
- Config – otvorí okno, v ktorom sa nastavujú parametre pre všetky časti experimentu,
- Set Config – spustí sa aplikácia a pre každú časť sa zistí, či sú zadané parametre v správnom tvare a prebehnú všetky ostatné kontroly (či je pripojený a zapnutý zosilňovač...),
- Start – spustí samotný experiment. Po stlačení sa tlačidlo premenuje na *Suspend* a ním sa dá potom prebiehajúci experiment ukončiť. Po skončení sa premenuje na *Resume* - funkcionalita rovnaká ako pri Start,
- Quit – vypne celú aplikáciu.

2.2 Dizajn BCI2000

Aplikácia BCI2000 je postavená na štandardnom modeli BCI, ktorý som popisoval v časti 1.3. Tento model (pre ilustráciu pozri zjednodušený model na obr. 2.1) pozostáva zo štyroch modulov, ktoré medzi sebou komunikujú:

- Source (skladá sa z Data Acquisition a Storage) - získavanie dát zo zosilňovača a ich ukladanie,
- Signal Processing - spracovanie získaného signálu,
- User Application - aplikácia, ktorá používa výstupy zo Signal Processing a väčšinou ich vizualizuje (spätná väzba pre používateľa),

- Operator - mozog celej aplikácie, zabezpečuje komunikáciu predchádzajúcich modulov.



Obr. 2.1: Dizajn aplikácie BCI2000 pozostávajúci zo štyroch modulov: Operator, Source, Signal Processing a Application[1].

Moduly sú nezávislé aplikácie, ktoré medzi sebou komunikujú prostredníctvom protokolu založenom na TCP/IP. Tento protokol dokáže prenášať všetky potrebné informácie (signály, hodnoty premenných...) a je navrhnutý tak, aby ho nebolo treba meniť, ak sa zmení niektorý z modulov.

Signály z mozgu sú spracovávané synchrónne v blokoch, ktoré obsahujú konštantný počet vzoriek. Jednu vzorku predstavujú hodnoty namerané na elektródach, ak napr. meriame 8 kanálov, tak vzorka je vektor s 8 hodnotami. Takýto blok dát vzniká v module Source získaním dát zo zosilňovača. Ten ho posiela do modulu Signal Processing, ktorý z dát vyextrahuje príznaky a preloží ich na riadiace signály, ktoré potom ovládajú modul User Application. Ten pošle informácie o udalostiach späť do modulu Source, ktorý ich spolu so signálmi z mozgu uloží na disk. To nám umožňuje rekonštrukciu celého merania pri off-line analýze. Celý tento proces „stráži“ modul Operator, ktorý zabezpečuje tok dát medzi jednotlivými modulmi, sprístupňuje im globálne premenné a vykresľuje výstupy.

Voľba veľkosti bloku závisí od toho, aké máme možnosti pri spracovávaní signálu (výpočtová sila) a ako veľmi je pre nás dôležité časovanie. V on-line systéme vytvorenom

[1]Všetky obrázky používané v tejto kapitole sú prevzaté zo Schalk and Mellinger (2010) alebo zo stránky http://www.bci2000.org/wiki

v BCI2000 je pre nás dôležité určiť, ako často chceme prezentovať stimul užívateľovi. Predstavme si ovládanie kurzoru na obrazovke pomocou BCI. Zosilňovač získa nejaké signály, tie sa spracujú, vyhodnotia a aplikácia posunie kurzor nejakým smerom (vykreslenie kurzoru je vlastne prezentácia stimulu pre užívateľa). To, ako často za sekundu sa kurzor posunie, závisí od zvolenia veľkosti bloku dát a vzorkovacej frekvencie (koľkokrát za sekundu získa zosilňovač dáta z meraných elektród). Z toho by sa mohlo zdať, že čím menšia veľkosť bloku, tým lepšie. Problém je v tom, že blok dát musí „precestovať“ celý cyklus od získania zo zosilňovača cez jeho spracovanie až po uloženie dát na disk. Na to treba nejakú výpočtovú silu a nejaký čas. Keď chceme, aby náš BCI fungoval v reálnom čase, tak treba zabezpečiť aby čas, za ktorý sa spracuje jeden blok, bol kratší ako dĺžka bloku. Je neprijateľné, aby v čase, keď je nejaký blok ešte stále spracovávaný v moduli Signal Processing, bol ďalší vytvorený a „poslaný na cestu“ v cykle BCI2000. Väčšinou sa pri meraní do 64 kanálov používa vzorkovacia frekvencia 256Hz a veľkosť bloku 8 vzoriek, čo zabezpečí v bežných prípadoch dostačujúcu obnovovaciu frekvenciu 32Hz (32 krát za sekundu sa napr. posunie kurzor).

Ako som už spomínal, moduly medzi sebou komunikujú prostredníctvom protokolu, vďaka ktorému je možné kedykoľvek zmeniť jeden modul určitého typu za druhý. Je to zabezpečené tým, že informácie, ktoré sú prenášané medzi nimi sú vysoko štandardizované a tým sa minimalizuje ich závislosť. Už pri návrhu modulov bola každá BCI fukcionalita umiestnená do modulu, do ktorého logicky patrí. Napr. celý cyklus toku bloku informácií začína v moduli Source, tak sa tento modul správa ako systémový časovač. Podobne, keďže vlastnosti spätnej väzby (napr. jej dĺžka) súvisia s modulom User Application, tak sa nachádzajú presne tam. Preto je možné naprogramovať si vlastný modul Signal Processing a nahradiť ním ten pôvodný v distribúcii (za predpokladu že používa správne vstupy a generuje stanovené výstupy).

2.2.1 Systémové premenné

BCI2000 používa tri druhy systémových premenných:

- parametre,
- označovač udalostí (event markers),
- signály.

Systémové parametre sú také premenné, ktoré sa nemenia počas jedného merania (napr. špecifikovaná dĺžka online spätnej väzby). Na druhú stranu označovače uda-

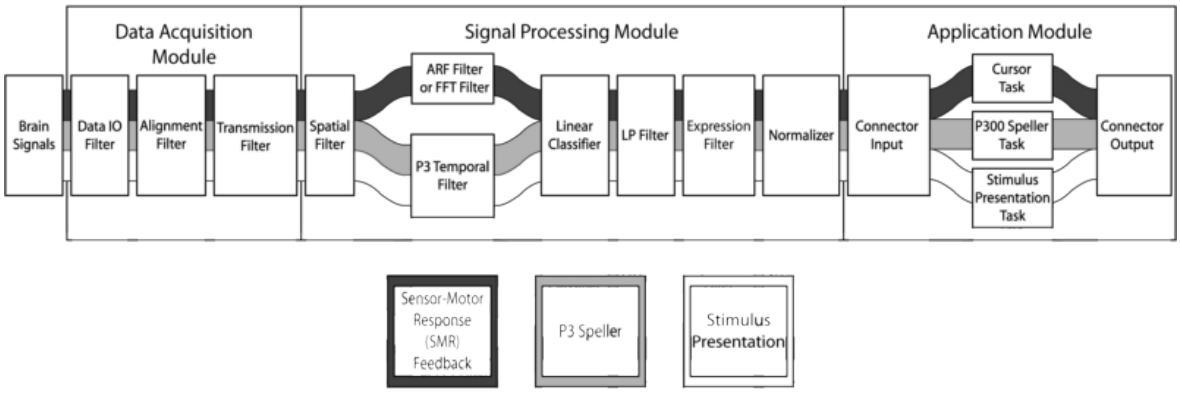
lostí zaznamenávajú, kedy a aké udalosti nastali počas daného merania (napr. na obrazovke sa zobrazil stimul číslo 1, teraz zmizol, začala sa spätná väzba...). Takéto informácie nám označujú v dátach kedy nastali nejaké významné udalosti a to nám pri offline analýze umožňuje rekonštruovať meranie a sledovať, ako sa zmenil signál, keď nastala konkrétna udalosť. Tieto značky môže vytvárať, modifikovať alebo monitorovať ľubovoľný z modulov. Systémové signály sú funkcie signálu získaného z mozgu užívateľa, ktoré sú získavané a modifikované jednotlivými modulmi.

Každý modul môže požiadať modul Operator, aby mu vytvoril ľubovoľný počet systémových parametrov (rôznych typov ako napr. čísla, vektory, matice alebo reťazce), alebo označovačov udalostí (s dĺžkou 1 až 16 bitov). Napr. modul Source môže požiadať o parameter, ktorý definuje vzorkovaciu frekvenciu. Tento parameter ostane konštantný počas celého merania, bude dostupný ostatným modulom a automaticky bude uložený spolu s nameranými dátami. Podobne, modul Signal Processing, ktorý má detekovať nevhodné artefakty (napr. tie spôsobené žmurkaním), môže potrebovať označovač udalostí, ktorým má v dátach zaznamenať, kedy daný šum nastal. Modul *User Application* môže požadovať označovač udalostí nato, aby označil, kedy aký stimul prezentoval užívateľovi. Tieto rozličné systémové premenné sú automaticky posielané medzi modulmi, a uložené na začiatok súboru pred samotný signál z mozgu.

2.3 Filtre

V časti 2.2 sme predstavili model BCI2000 skladajúci sa zo štyroch základných modulov (Core modules). Ak však chceme niečo v tejto aplikácii upraviť alebo doprogramovať, tak je potrebný podrobnejší pohľad. Tok dát v BCI2000 (po blokoch) si môžeme predstaviť ako vodovodnú rúru. Na jej začiatku sú signály z mozgu získané zo zosilňovača a na konci je aplikácia, ktorá dáva spätnú väzbu. Medzi nimi je súbor malých aplikácií, ktoré majú svoj vstup, niečo s ním spravia (pretransformujú ho, upravia) a dajú ho na výstup. V kontexte BCI2000 im hovoríme filtre. Všetky základné moduly (Core modules) okrem Operator modulu sa skladajú z filtrov. Môže ich byť ľubovoľný počet, ale každý modul musí obsahovať aspoň jeden filter. Zoznam a usporiadanie filtrov v štandardnej distribúcii BCI2000 verzie 3.0.5 možno vidieť na obr. 2.2.

Už sme spomínali, že BCI2000 môžeme používať na dve základné paradigmy BCI - P300 a ERD (obe popísané v časti 1.3.2). Pre paradigmu P300 používame hláskovač, ktorý sa v BCI2000 volá *P3 Speller*. Pomocou paradigmy ERD ovládame aplikáciu



Obr. 2.2: Filtre v BCI2000 a tok informácií pre rozličné aplikácie (Cursor Task, P300 Speller Task, Stimulus Presentation Task).

Cursor Task (ovládanie loptičky na obrazovke). Z predchádzajúceho a obr. 2.2 vyplýva, že BCI2000 poskytuje tri rôzne moduly typu User Application (tri rôzne aplikácie):

- Stimulus Presentation (Prezentácia stimulov),
- P3 Speller (P3 hláskovač - pozri časť 1.3.2),
- Cursor Task (Kurzorová úloha založená na ERD - pozri časť 1.3.2).

Podľa toho, aký experiment, alebo akú BCI paradigmu chceme robiť, tak si zvolíme druh filtrov pre svoju aplikáciu. Z obr. 2.2 je zrejmé, že väčšinu filtrov majú aplikácie rovnaké a líšia sa hlavne v moduloch Signal Processing a User Application. Teraz si popíšeme podrobnejšie jednotlivé moduly a ich najdôležitejšie filtre. Funkciu filtrov budeme vysvetľovať väčšinou na našom zosilňovači a aplikácii *Cursor Task*. Taktiež budeme presnejšie popisovať transformáciu dát zo vstupov na výstupy v jednotlivých filtroch.

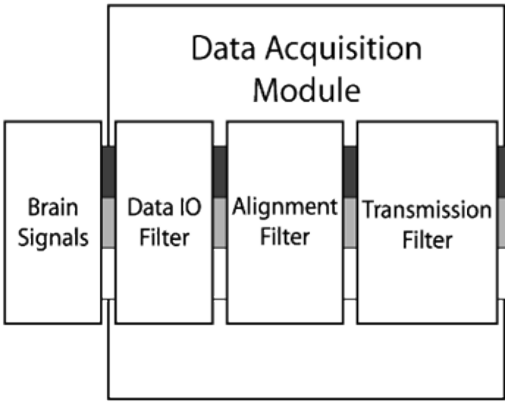
2.3.1 Source module

Úlohou tohto modulu je získavať mozgové signály zo zosilňovača a posúvať kalibrovaný signál do modulu *Signal Processing*. Pozostáva z časti, ktorá získava samotný signál (Data Acqisition) a druhej, ktorá získané dáta ukladá na disk v rôznych formátoch (formát BCI2000, EDF - formát používaný pri výskume spánku a GDF - variant EDF vyvinutý pre BCI aplikácie). Komponent na získavanie dát má rôzne implementácie (aby vedel komunikovať s rôznymi zosilňovačmi). Medzi prvými boli implementácie pre tri zosilňovače od firmy g.tec, a to g.USBamp, g.MOBIlab, g.MOBIlabPlus, ktoré sú sučasťou štandardnej distribúcie.

Okrem nich je implementovaný modul Signal Generator, ktorý sa používa na testovanie a správnu konfiguráciu vytvoreného BCI systému. Produkuje náhodný signál, ktorý sa skladá zo sínusoíd, ktorých frekvencia sa dá meniť polohou myši (platí to iba pre jeden kanál). Používali sme ho hlavne pri implementácii vlastých filtrov na otestovanie, či sú všetky parametre správne nastavené a či filter vykonáva požadovanú funkciu. Zosilňovač väčšinou používa pre svoju prácu viacero ľudí a nie vždy je možné mať ho počas implementovania nového filtra alebo modulu pripojený. Existuje však aj množstvo iných, ktoré sú určené pre produkty výrobcov ako BioSemi, BrainProducts, Cleveland Medical Devices, Data Translation, Electrical Geodesics, Measurement Computing, National Instruments, Neuroscan, OpenEEG alebo Tucker-Davis Technologies. Tieto implementácie naprogramovala komunita užívateľov BCI2000 a nenachádza sa v štandardnej distribúcii. Spolu je podporovaných viac ako 15 rôznych zosilňovačov. Na to, aby aplikácia mohla používať nejaký vybraný zosilňovač, ktorý zatiaľ nie je podporovaný, stačí iba naimplementovať filter, ktorý z neho bude získavať dáta a posielať ich ďalej v správnom formáte.

Formát dát, v ktorom BCI2000 štandardne ukladá dáta, pozostáva z hlavičky, v ktorej sú uložené hodnoty parametrov všetkých používaných filtrov a binárne zakódovaného signálu so značkovačmi udalostí. Tento formát dokáže uložiť ľubovoľne veľký počet kanálov, parametrov a značkovačov udalostí a tým poskytuje protokol z ľubovoľne zložitého experimentu.

Tento modul používa tri základné filtre: Data IO Filter, Alignment Filter a Transmission Filter (pozri obr. 2.3).



Obr. 2.3: Filtre v moduli Source (presnejšie jeho časti Data Acquisition).

Data IO Filter

Tento filter je časťou každého Source modulu. Vykonáva štyri základné činnosti: získavanie dát zo zosilňovača, ich kalibrácia do fyzikálnych jednotiek (napr. μV), ukladanie a vizualizácia získaného signálu. Pre získavanie signálov tento filter implementuje rozhranie pre ADC komponent, ktorý je v každom Source moduli (časť komunikujúca priamo so zosilňovačom). Pre ukladanie získaných dát používa jeden z mnohých FileWriter komponentov, ktorý ukladá dáta vo zvolenom formáte.

Niektoré parametre používané v tomto filtri sú závislé od použitého zosilňovača, pričom ostatné ostávajú všade rovnaké. Medzi tie spoločné patrí počet meraných kanálov (koľko elektród používame), veľkosť bloku dát (koľko vzoriek sa má spojiť do jedného bloku a naraz spracovávať), vzorkovacia frekvencia (koľkokrát za sekundu má zosilňovač získať údaje z elektród), alebo pomenovanie meraných kanálov (nepovinné, ale zlepšuje prehľadnosť, pre EEG merania by mali súhlasiť so systémov 10-20). Pre príslušný FileWriter určujeme priečinok, do ktorého sa majú dáta ukladať (absolútna alebo relatívna cesta vzhľadom na priečinok prog obsahujúci spustiteľné súbory), meno subjektu, číslo sedenia (session) a číslo merania (run). Vo zvolenom priečinku sa vytvorí ďalší s názvom subjektu, ku ktorému je pridané čisilo sedenia (my sme pri mene subjektu používali iba iniciály, preto sa napr. vytvoril priečinok MK001 potom MK002 atď). Do vytvoreného priečinku sa potom vytvorí súbor pre každú sadu pokusov zvlášť. V našom prípade sa vytvoril napr. MKS002R01.dat, čo znamená subjekt MK, sedenie druhé a prvá sada pokusov. Okrem týchto súborov sa vytvorí súbor typu *.applog, ktorý zaznamenáva informácie o jednotlivých sadách pokusov, ich výsledky atď.

Pre nami používaný zosilňovač g.USBamp sme museli nastaviť zoznam kanálov, ktoré chceme merať, počet kanálov na zariadenie (keby sme mali viac zosilňovačov, tak ich vieme pospájať a tým získať viac kanálov), akého typu má byť signál (integer alebo float), alebo nastavenie uzemnenia a referenčnej bázy. Zariadenie taktiež ponúkalo aplikovanie hardvérových filtrov. Aplikovaním horného a dolného filtra sme mohli nastaviť rozsah merania (napr. 0.1–60Hz) a taktiež sme mohli aplikovať filter na odstránenie frekvencie, ktorou sa prenáša elektrický prúd v našej sieti (tzv. notch filter). Pri všetkých filtroch sa nastavuje frekvencia a ich typ (Butterworth, Chebyshev...).

Okrem spomínaných parametrov sa dá nastaviť potreba vizualizácie nameraných signálov, prípadne dĺžka úseku a jeho rozsah.

Alignment Filter

Zosilňovače získavajú dáta buď súbežne pre všetky kanály naraz, alebo postupne jeden kanál za druhým. V druhom prípade vzniká časový posun medzi jednotlivými kanálmi. Tento časový posun odstraňuje práve Alignment Filter pomocou lineárnej interpolácie medzi nameranými hodnotami. Takýto problém mávajú hlavne staršie zosilňovače, nové už získavajú dáta naraz. Pri našom hardvéri sme takýto problém nemali. Ako parameter sa udáva informácia, ktorý kanál o koľko posunúť.

Transmission Filter

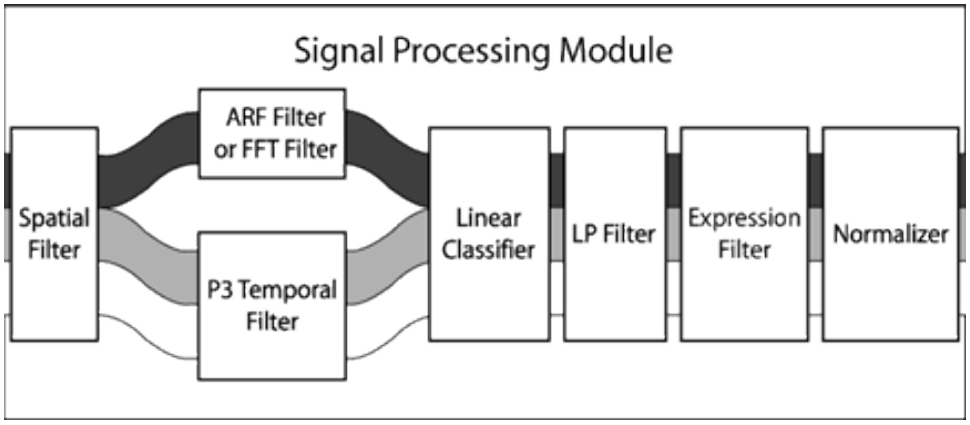
Tento filter nerobí nič iné, iba vyberie zo vstupných kanálov tie, ktoré chceme dalej spracovávať a pošle ich na výstup. Je to väčšinou posledný filter Source modulu, kde vyberáme, ktoré kanály budeme reálne využívať pri spracovaní signálu, čo zmenšuje veľkosť dát, ktoré sa musia spracovávať. Ako parameter nastavujeme zoznam kanálov, ktoré sa majú dostať na výstup.

2.3.2 Signal Processing module

Modul Signal Processing dostáva z modulu Source signály z mozgu, ktoré konvertuje na riadiace signály, ktorými sa potom ovláda koncová aplikácia. Z pohľadu BCI môžeme rozdeliť túto konverziu na dve časti: extrakcia príznakov (feature extraction) a ich preklad na riadiace signály (translation). Túto konverziu zabezpečuje skupina filtrov (pozri obr. 2.4), ktoré sú nezávislé od seba, iba konvertujú svoj vstup na výstup. Vďaka tomu ich môžeme ľubovoľne kombinovať a usporiadavať. Treba samozrejme zabezpečiť logický sled filtrov a tým správny tok dát.

V prípade aplikácie *Cursor Task* je vstupom do modulu signál z mozgu a výstupom budú riadiace signály (ich počet závisí od toho, koľko rozmerov kurzoru chceme ovládať). Riadiaci signál v tomto prípade tvoria čísla s nulovým priemerom a jednotkovým rozptylom.

V časti exstrahovania príznakov momentálne BCI2000 poskytuje dva základné filtre. Prvým je *priestorový filter* (spatial filter), ktorý vytvorí výstup nejakou lineárnou kombináciou vstupov pomocou definovanej matice (spatial matrix). Druhý môžeme nazvať *časový filter* (temporal filter). V distribúcii BCI2000 sa nachádzajú momentálne tri rôzne variácie takéhoto filtra: autoregresná spektrálna analýza, spektrálna analýza založená na FFT (Fast Fourier Transform - Fourierová transformácia) a filter, ktorý robí priemer vyvolaných reakcií (napr. pre P300 - pozri časť 1.3.2).



Obr. 2.4: Filtre v moduli Signal Processing.

V druhej časti sa získané príznaky prekladajú do riadiacich signálov, ktoré sú nezávislé od koncovej aplikácie. To vykonáva tiež dvojica filtrov. Prvý aplikuje lineárny klasifikátor a druhý normalizuje výstupné signály tak, aby mali priemer 0 a vybraný rozptyl zadanej hodnoty (napr. 1). Výstup normalizéra je väčšinou výstup celého modulu Signal Processing.

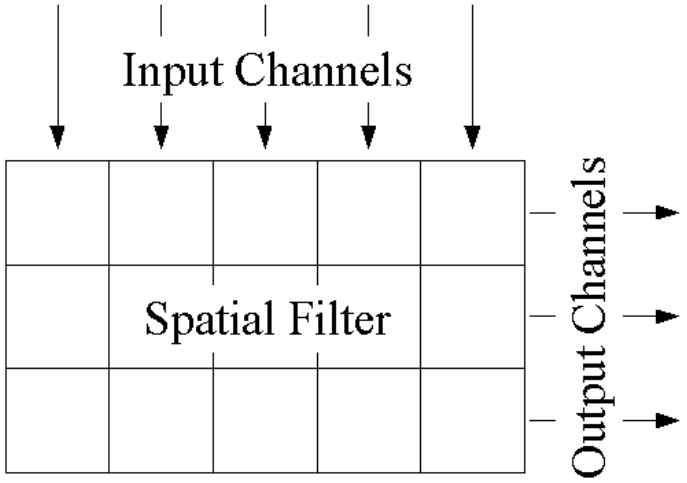
Okrem týchto filtrov BCI2000 poskytuje ešte LP Filter a Expression Filter, ktorých funkčnosť popíšem neskôr.

Spatial Filter

Spatial filter vracia na výstup lineárnu kombináciu svojich vstupov. Väčšinou sú jeho vstupom nefiltrované signály z mozgu poskytnuté Source modulom. To, aká transformácia je aplikovaná, je definované v priestorovej matici (spatial matrix). Táto matica má rozmery M*N, kde M je počet stĺpcov a je rovný veľkosti vstupu (koľko kanálov meriame) a N je počet riadkov reprezentujúci veľkosť výstupu (počet kanálov, ktoré chceme mať na výstupe). Pre lepšiu predstavu pozri obr. 2.5.

Existuje viacero štandardných priestorvých filtrov, ktoré ponúka aj BCI2000:

- veľký Laplacian (Large Laplacian) - od danej elektródy odčítame násobky (dokopy rovné jednej) jej vzdialenejších susedov. Napr. hodnota elektródy C3 bude nasledovná: C3 - 0,25*Cz - 0,25*P3 - 0,25*T7 - 0,25*F3 (kedže má štyroch susedov, tak odčitujeme 0,25-násobok),
- malý Laplacian (Small Laplacian) - od danej elektródy odčítame násobky jej najbližších susedov. Napr. hodnota elektródy C3 bude nasledovná: C3 - 0,25*FC3 - 0,25*C1 - 0,25*CP3 - 0,25*C5,



Obr. 2.5: Vizualizácia funkcie priestorového (Spatial) filtra.

- CAR (Common Average Reference) - vypočíta priemer všetkých kanálov a ten odpočíta od vybraných elektród (dá sa nastaviť parametrom).

Každý z typov používa iný algoritmus na počítanie lineárnej transformácie a môže mať dopad na zaťaženie procesora a tým pádom na celkový výkon BCI systému. Ako parameter tohto filtra zadávame typ priestorovej matice:

- úplná,
- riedka (sparse).

Pri všetkých typoch matíc stĺpce reprezentujú vstupné kanály (M) a riadky výstupné (N). Každý riadok znamená, akú lieárnu kombináciu vstupov chceme na výstup.

Ak použijeme úplnú maticu, tak je počet výstupov a vstupov rovnaký. Ak by sme chceli maticu, ktorá nezmení vstup, tak stačí zadať jednotkovú maticu (jednotky na diagonále a na ostatných miestach nuly). Časová zložitosť takéhoto riešenia je *O(MN)*, čo sa pri vysokom počte meraných kanálov môže negatívne prejaviť na výkone.

Pomocou riedkej matice definujeme vzťah medzi vstupnými a výstupnými kanálmi s určitou váhou. Matica v tomto prípade obsahuje tri stĺpce a jeden riadok pre definíciu každého vzťahu. V prvom stĺpci sa uvádza vstupný kanál, v druhom výstupný a v treťom váha, ktorou sa prenásobí vstupný kanál pred tým, ako sa pričíta k výstupnému. Tento typ sa odporúča v prípade merania veľkého počtu kanálov. Ak by sme použili úplnú maticu, tak by sa výstup tvoril zo všetkých vstupov (aj keby sme ich iba násobili nulou). Ak máme napr. 64 elektród a výstup je tvorený iba násobkom štyroch elektród, tak použitím riedkej matice by sme ušetrili 60 násobení. Keďže je N≤M, tak

je časová zložitosť tohto typu niekde medzi *O(M)* a *O(MN)*. V najhoršom prípade však $O(M^2)$.

AR Filter

AR Filter počíta autoregresný model svojich vstupov (väčšinou výstup zo Spatial filtra) pomocov metódy maximálnej entrópie (Burgova metóda). Výpočet prebieha pre každý vstup samostatne. Výstupom filtra môžu byť AR koeficienty, alebo odhadované spektrum výkonu jednotlivých častí (spektrálna analýza po jednotlivých častiach). Preto môže byť použité namiesto FFT (rýchla Fourierova transformácia).

V časti 1.2.2 sme uvideli, že výsledok spektrálnej analýzy má na jednej osi frekvencie a na druhej ich „silu“. AR filter však nepracuje s frekvenciami ako takými, rozdeľuje celé frekvenčné pásmo na menšie časti (biny) rovnakej šírky. Preto pri nastavovaní treba ako prvé zadať, pre aké časti (biny) sa bude spektrálna analýza počítať. Pomocou parametrov FirstBinCenter a LastBinCenter nastavíme stredy prvej a poslednej časti (v Hz). Ich šírku nastavujeme parametrom BinWidth (v Hz).

Pri našich experimentoch sme zvykli pracovať s pásmom 0–30Hz, pričom šírka jednej časti (binu) bola 2Hz. Prvá časť teda reprezentovala frekvenčné pásmo od 0Hz do 2Hz, druhá časť 2–4Hz atď. Takže ako FirstBinCenter sme nastavili hodnotu 1Hz, LastBinCenter bol 29Hz a BinWidth 2Hz. Tým sme dostali spolu 16 binov. Za zmienku stojí ešte parameter EvaluationPerBin, ktorý hovorí o tom, na koľko rovnakých častí sa má pri výpočte jednotlivá časť frekvenčného pásma (bin) rozdeliť (bežnou hodnotou je 10). Výslednou hodnotou je potom priemer hodnôt tých malých častí.

Ďalším dôležitým parametrom je WindowLength, ktorý udáva z akého veľkého okna sa bude počítať (napr. 500ms). V praxi to vyzerá tak, že v danom momente sa zoberie signál nastavenej dĺžky smerom do minulosti. Keďže je to autoregresívny model, tak sa dá ešte nastaviť, akého rádu má byť (táto hodnota približne zpodpovedá najväčšiemu počtu lokálnych maxím, ktoré môže výsledná spektrálna analýza mať) a či výstupom má byť spektrálna analýza alebo iba AR koeficienty.

Výstupom je teda matica M*N, kde M je počet meraných kanálov a N je počet častí frekvenčného pásma (binov). Hodnota matice určuje akú „silu“ sme namerali na kanáli M a frekvencii (bine) N.

FFT Filter

Tento filter aplikuje rýchlu Fourierovu transformáciu na nejaké okno signálu (napr. 500ms) na vybraných kanáloch svojho vstupu a výstupom je spektrálna analýza. V BCI-2000 sa väčšinou používa ako alternatíva *AR Filtra*.

Jeho parametrami je veľkosť a typ okna, v ktorom sa počíta (Hammingovo, Hannovo alebo Blackmanovo okno) a zoznam kanálov, na ktorých sa má analýza počítať. Taktiež sa dá nastaviť, aby sa výsledky vizualizovali priamo užívateľovi.

FFT Filter na výpočet používa knižnicu FFTW3, ktorú si však treba zaobstarať zvlášť, lebo nie je súčasťou inštalácie BCI2000.

P3Termporal Filter

Úlohou tohto filtra je zoskupovať a priemerovať dáta po epochách pri experimentoch založených na online klasifikácii odoziev vyvolaných udalosťou (ERP - Event related potentials). Pri experimente sa užívateľovi zobrazuje určitý počet stimulov a systém má zisťovať, aká bola jeho reakcia. Využíva sa napr. v aplikáci *P300 Speller* popísanej v časti 1.3.2. Filter zoskupuje dáta po epochách pre každý stimul samostatne. Po určitom nastavenom počte epoch filter vypočíta časový priemer a na výstup vráti výsledný priebeh signálu.

Výstup P3Temporal filtra je väčšinou posielaný do lineárneho klasifikátora, ktorý zoberie dáta z viacerých kanálov a lineárne ich skombinuje do jedného výstupu. Toto číslo reprezentuje veľkosť vyvolanej odozvy. Najčastejšie sa používa s aplikáciami *Stimulus Presentation* a *P3 Speller*.

Jeho najdôležitejšími vstupnými parametrami sú dĺžka jednej epochy a počet epôch, z ktorých má počítať priemer. Okrem toho sa dá zapnúť vizualizácia jeho výstupov.

Linear Classifier

AR Filter, *FFT Filter* alebo *P3Temporal Filter* (alebo nejaký vlastný naprogramovaný) extrahujú zo signálu (ktorý predtým prešiel *Spatial Filtrom*) nejaké príznaky. Úlohou filtra Linear Classifier (lineárny klasifikátor) je prekladať tieto príznaky do riadiacich signálov, ktoré potom ovládajú koncovú aplikáciu. Táto transformácia prebieha podľa zadaných lineárnych rovníc. Každý riadiaci signál je teda nejakou lineárnou kombináciou vstupov. Vstup do tohto filtra má dva rozmery, počet kanálov (N) $\times$ počet elementov(M) na danom kanáli a výstupom je vektor, ktorého veľkosť zodovedá počtu riadiacich signálov. Výstupy sa počítajú pomocou nasledovnej rovnice:

$$output_k = \sum_{i=1}^{N} \sum_{j=1}^{M} input_{ij} Classifier_{ijk} \tag{2.1}$$

Výraz $Classifier_{ijk}$ v rovnici 2.1 reprezentuje vzťah medzi vstupom a výstupom. Určený je klasifikačnou maticou, ktorá sa zadáva ako jediný parameter tohto filtra (jeho názov je Classifier).

V našej práci sme používali BCI, ktoré bolo založené na desynchronizácii μ-rytmu a preto bola vstupom do lineárneho klasifikátora matica, ktorá obsahovala hodnoty spektrálnej analýzy pre všetky kanály a príslušné časti spektrálneho pásma (biny, vysvetlené pri popise AR Filtra). Výstupom boli riadiace signály, ktoré po normalizácii vo filtri *Normalizer* (pozri nižšie) ovládali pohyb kurzora po obrazovke (aplikácia *Cursor Task*). Museli sme však definovať, ako sa má informácia o spektre preložiť do riadiacich signálov. To je dané v už spomínanom parametri Classifier. Je to matica, ktorá má štyri stĺpce a každý jej riadok vyjadruje určitý vzťah vstupu s výstupom. Prvý stĺpec reprezentuje vstupný kanál, druhý vstupný element, tretí výstupný kanál (riadiaci signál) a štvrtý váhu s akou sa daný vzťah použije.

Pri našich experimentoch sme na vstupe mali všetky kanály a všetky biny, pričom nie všetky sú pre nás zaujímavé. Pre každý subjekt sme si pomocou offline analýzy zistili, ktoré príznaky sú pre daný subjekt a daný experiment významné (viac v časti 4.3) a len tie chceme, aby ovplyvňovali riadiaci signál. Napríklad sme zistili, že na ovládanie kurzora má daný subjekt najvhodnejšiu elektródu C4 a na nej na biny číslo 6 (reprezentuje pásmo 10–Hz) a 11 (pásmo 20–22Hz). Ďalej elektródu C3 a na nej biny číslo 5 a 11. V tomto prípade je klasifikačná matica popísaná v tabuľke 2.1.

Input channel	**Input element**	**Output channel**	**Weight**
C4	6	2	0,3
C4	11	2	0,7
C3	5	1	0,5
C3	11	1	0,5

Tabuľka 2.1: Príklad klasifikačnej matice pre Cursor Task.

Pri ovládaní kurzora v 2D potrebujeme dva riadiace signály, jeden na ovládanie horizontálneho a druhý na ovládanie vertikálneho smeru. Preto je interpretácia tabuľky 2.1 nasledovná: ako riadiaci signál číslo 1 zober výsledok spektrálnej analýzy z elektródy C4 a jej šiesteho binu prenásobený váhou 0,3 a k tomu pripočítaj výsledok spektrálnej

analýzy pre elektródu C4 a jej jedenásty bin prenásobený číslom 0,7. Druhý riadiaci signál sa počíta z elektródy C3 a jej binov 5 a 11, pričom obidve hodnoty prenásobíme číslom 0,5.

LP Filter

LP Filter je jednoduchý jednopólový časový nízkofrekvenčný filter s časovou konštantou T. Definovaný je rovnicami 2.2 a 2.3 (kde t je index vzorky proporcionálne v čase).

$$S_{out,0} = (1 - e^{-1/T})S_{in,0} \tag{2.2}$$

$$S_{out,t} = e^{-1/T}S_{out,t-1} + (1 - e^{-1/T})S_{in,t} \tag{2.3}$$

Z rovníc vyplýva, že výstup v čase nula ($S_{out,0}$) sa počíta ako vstup v čase nula prenásobený konštantou $1 - e^{-1/T}$ závislou od parametra T. Výstup v inom čase sa počíta ako hodnota v predchádzajúcom kroku ($S_{out,t-1}$) prenásobená konštantou ($e^{-1/T}$) a k tomu sa pričíta momentálny vstup opäť vynásobený konštantou ($1 - e^{-1/T}$)$S_{in,t}$.

Väčšinou sa používa na odstránenie vysokofrekvenčného šumu, ktorý môže produkovať lineárny klasifikátor. Jeho úlohou bolo, aby jeho výstupy neboli veľmi rozdielne. V našom prípade ovládania kurzora zabezpečoval, aby jeho smer nemenil veľmi rýchlo, ale aby zmena bola plynulá. To, akú veľkú váhu má mať výstup z predchádzajúceho kroku, sa nastavuje práve parametrom T (napr. 0.5s). V prípade, že sa nastaví na hodnotu 0, tak filter iba kopíruje svoj vstup na výstup.

Expression Filter

Tento filter používa aritmetické výrazy na premenu vstupu na výstup. Tieto aritmetické výrazy môžu obsahovať stavové premenné systému a tým môžme:

- upravovať spracovanie dát vzhľadom na stav aplikácie BCI2000,
- vložiť informáciu o stave ako ďalší kanál signálu,
- nahradiť spracované výsledky informáciou o stave.

Jeho vstupom je matica obsahujúca jednotlivé aritmetické výrazy, ktoré definujú výstup filtra. Jej riadky zodpovedajú kanálom vstupného signálu a stĺpce jednotlivým elementom. Napr. matica

$$\begin{matrix} Signal(1,1)^2 * (ResultCode == 0) & Signal(1,2)^2 * (ResultCode == 0) \\ Signal(2,1)^2 * (ResultCode == 0) & Signal(2,2)^2 * (ResultCode == 0) \end{matrix}$$

nahradí hodnoty signálu ich druhou mocninou a ak je premenná ResultCode rovná nule, tak je výstupom 0. Pri našich experimentoch sme tento filter nevyužívali.

Normalizer

Tento filter aplikuje lineárnu transformáciu na vstup (väčšinou riadiace signály produkované lineárnym klasifikátorom) tak, aby bol výstup v určitom rozsahu. Každý vstupný kanál je prenásobený určitým číslom (gain) a od výsledku sa potom odčíta iná hodnota (offset). Tieto hodnoty sa nastavujú v parametroch NormalizerOffset a NormalizerGain, ktoré sú definované pre každý vstupný kanál zvlášť. Lineárna transformácia potom prebieha podľa rovnice 2.4 (index i určuje poradie vstupu).

$$output_i = (input_i - \mathsf{NormalizerOffset_i}) \times \mathsf{NormalizerGain_i} \tag{2.4}$$

Dané dva parametre môžeme nastaviť manuálne dopredu, alebo ich môže BCI2000 počas behu experimentu automaticky počítať. Ak je táto možnosť zapnutá, tak filter počíta gain a offset na základe doteraz nameraných dát tak, aby výstupný signál mal nulový priemer a rozptyl rovný 1. Na to používa zásobníky, ktoré do seba podľa definovaných pravidiel ukladajú dáta. Tieto pravidlá sa definujú v parametri BufferConditions a sú v podobe logických výrazov. Hodnotou parametra je matica, pričom každý jej stĺpec reprezentuje jeden vstupný signál (ak používam dva riadiace signály, tak táto matica bude mať dva stĺpce) a každý riadok vyjadruje jeden zásobník. Hodnotou v matici je logický výraz, ktorý ak je vyhodnotený ako pravda, tak sa hodnota príslušného riadiaceho signálu uloží do daného zásobníka. Dĺžka všetkých zásobníkov je určená parametrom BufferLength (určená buď v počte vzoriek alebo v sekundách). Príklad takýchto podmienok pre zásobníky je v tabuľke 2.2.

Počas našich experimentov sme ovládali pomocou BCI jeden smer kurzora (vertikálny) a druhý (horizontálny) bol konštantný. Preto sme potrebovali dva riadiace signály a preto má matica v tabuľke 2.2 dva stĺpce. Prvý stĺpec obsahuje 0, lebo pre daný kanál sme nepotrebovali žiadny zásobník. V porovnaní s tým sme pre druhý

Control signal 1	Control signal 2
0	(Feedback) && (TargetCode==1)
0	(Feedback) && (TargetCode==2)

Tabuľka 2.2: Príklad logických výrazov používaných ako parameter BufferConditions pre filter Normalizer.

kanál používali dva - jeden na dáta, ktoré boli namerané počas spätnej väzby keď bol zobrazený cieľ číslo 1 a druhý pre cieľ číslo 2.

Ako som už spomínal, dané hodnoty gain a offset sa môžu počas experimentu automaticky meniť (adaptovať). Pomocou parametra Adaptation nastavujeme pre každý kanál zvlášť, či chceme adaptáciu zakázať (hodnota 0), či chceme aby sa menila iba hodnota offset (kvôli nulovému priemeru, hodnota 1), alebo či sa má okrem priemeru zabezpečiť aj jednotkový rozptyl (hodnota 2). Väčšinou nechceme, aby sa hodnoty adaptovali v každom kroku počas experimentu. Radšej chceme, aby sa počítalo napríklad iba na konci jedného pokusu (menšia časť sady pokusov, u nás jedno navigovanie kurzora po obrazovke). To sa nastavuje posledným dôležitým parameterom UpdateTrigger. Jeho hodnotou je logický výraz, ktorého hodnota, keď sa zmení z False na True, tak sa spustí výpočet, ktorý zistí správne hodnoty gain a offset podľa dát vo všetkých zásobníkoch určených pre daný riadiaci signál. V našom prípade to bola podmienka ($Feedback == 0$), čiže keď skončí spätná väzba.

Na príklade skúsim presne vysvetliť priebeh funkcie tohto filtra. Predstavme si, že chceme ovládať kurzor v 1D (kontrolujeme iba jeho vertikálny smer) a na obrazovke máme dva možné ciele, jeden hore (označme ho 1) a druhý dole (označme ho 2). Počas spätnej väzby, keď sa kurzor pohybuje zľava doprava a my sa ho snažíme ovládať, sa vstupné riadiace signály ukladajú do zásobníka 1 alebo 2 podľa toho, ktorý z cieľov je zobrazený. Okrem toho je každý vstup prenásobený hodnotou parametra NormalizerGain a odčítaná je od neho hodnota parametra NormalizerOffset. Keď skončí spätná väzba (trafil som cieľ alebo nie), tak sa spustí adaptácia. Zoberú sa dáta z oboch zásobníkov a vypočíta sa akým číslom ich musím prenásobiť a aké číslo musím odčítať, aby všetky hodnoty mali dokopy nulový priemer a jednotkový rozptyl. Tieto hodnoty sa potom nastavia ako aktuálne parametre NormalizerGain a NormalizerOffset.

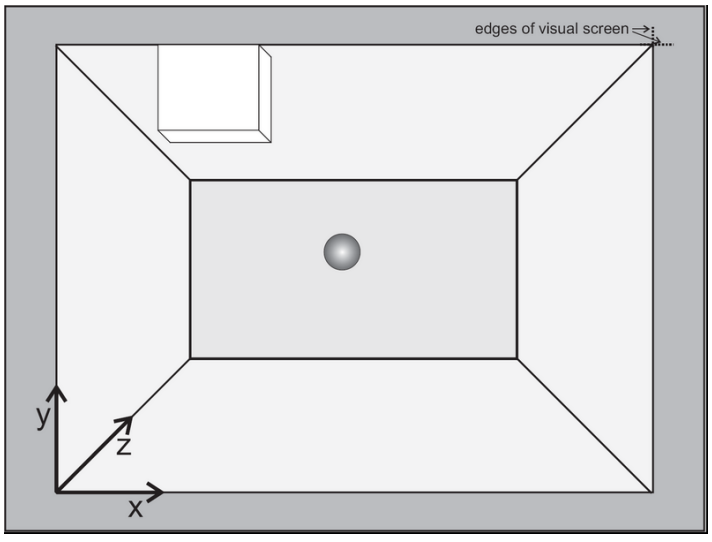
2.3.3 User Application Module

Tento modul dostáva riadiace signály z modulu *Signal Processing* a používa ich na ovládanie aplikácie. Súčasná verzia BCI2000 obsahuje tri takéto aplikácie, ktoré implementujú dve populárne paradigmy so spätnou väzbou: trojdimenzionálne ovládanie kurzoru (*Cursor Task*) a hláskovanie založené na P300 (P3 Speller). Posledná aplikácia sa používa pri prezentácii zvukových a vizuálnych stimulov užívateľovi (Stimulus Presentation).

Cursor Task

Cieľom tejto aplikácie je poskytnúť užívateľovi BCI2000 modul, pomocou ktorého môže vykonávať jeden zo základných experimentov určených pre BCI a to ovládanie kurzoru na obrazovke. Pomocou nej dokážeme vykonávať 1D, 2D alebo až 3D pohyb.

Používateľovi sa zobrazí scéna v tvare obdĺžnika (v 3D je to kocka) a kurzor je reprezentovaný guľou. Na obrazovke sa postupne zobrazjú ciele (targets) v podobe obdĺžnikov (v 3D sú to kvádre), ktoré má užívateľ kurzorom trafiť (pozri obr. 2.6). My sme používali najjednoduchší variant, ovládanie kurzoru v 1D. Kurzor sa pohyboval konštantnou rýchlosťou z ľavej časti obrazovky po pravý okraj, kde sa nachádzal cieľ buď v hornej alebo spodnej časti. Našou úlohou bolo ovládať vertikálny pohyb daného kurzoru a trafiť cieľ.



Obr. 2.6: Ukážka obrazovky v aplikácii Cursor Task pri trojrozmernom ovládaní kurzoru. Biela kocka reprezentuje cieľ a tmavá guľa kurzor.

Experiment pomocou tejto aplikácie sa skladá zo sady pokusov (run). Jeden pokus

je snaha užívateľa trafiť kurzorom cieľ. Pokus sa skladá zo štyroch fáz. V prvej fáze sa zobrazí prázdna scéna. Potom sa objaví cieľ umiestnený niekde v scéne. Objekt sa môže objaviť na lubovoľnej pozícii, ktorú užívateľ (dizajnér experimentu) nastaví (v našom prípade na pravom okraji hore alebo dole). Po tejto fáze sa objaví na scéne kurzor, ktorý sa hneď začne pohybovať podľa riadiacich signálov posielaných z modulu Signal Processing (to je fáza, počas ktorej prebieha spätná väzba). Pokus skončí, keď užívateľ trafí kurzorom cieľ alebo ak ho nestihne zasiahnuť v určenom časovom limite. Potom začína posledná fáza, ktorou je vyhodnotenie pokusu. Ak užívateľ zasiahne cieľ, tak sa zmení jeho farba pre zvýraznenie úspechu. Ak ešte nebol prekročený počet pokusov, tak aplikácia pokračuje opäť prvou fázou.

Poloha kurzora je určovaná pomocou troch riadiacich signálov. Každý z nich ovláda pohyb v jednom rozmere. Signál 1 zodpovedá súradnici X, 2 súradnici Y a 3 súradnici Z (súradnice sú vyznačené na obr. 2.6). Daný riadiaci signál sa skladá z jedného elementu, ktorým je hodnota, ktorá určuje rýchlosť pohybu kurzoru v danom smere. V našom prípade to boli hodnoty, ktoré mali nulový priemer a jednotkový rozptyl.

Táto aplikácia má veľa parametrov, ktoré sú pre prehľadnosť rozdelené do viacerých kategórií. Ako prvé sa môžu nastaviť vlastnosti okna, v ktorom sa bude zobrazovať výstup pre užívateľa. Treba určiť veľkosť okna a jeho polohu (ak má byť na externom monitori). Ďalej sa nastavujú vlastnosti samotného pokusu. Tu sme určovali dĺžky jednotlivých fáz pokusu (čas medzi pokusmi, čas pred spätnou väzbou, dĺžka trvania spätnej väzby a čas po trafení cieľa) a celkový počet pokusov v jednej sade pokusov.

Potom treba nastaviť vlastnosti cieľov. Definujeme ich počet, farbu, textúru a poradie v akom budú prezentované užívateľovi. Pre jednotlivé ciele nastavujeme pozíciu ich stredu a rozmery (pre všetky tri dimenzie). Podobné vlastnosti vieme definovať aj pre kurzor.

Stimulus Presentation Task

Úlohou tejto aplikácie je prezentovať užívateľovi vizuálne, alebo audiálne stimuly, alebo ich kombináciu. Pritom sa v dátach označujú časy, kedy boli stimuly prezentované, aby sa pri analýze dali spájať so zmenami v dátach. Namerané údaje sa potom používajú pri offline analýze, napríklad pre aplikáciu *Cursor Task* alebo *P300 Speller Task*.

Zobrazenie jedného stimulu označujeme ako sekvencia (sequence). Parametrami môžeme určiť, aké stimuly sa majú zobrazovať. Taktiež nastavujeme dĺžku sekvencie alebo časový interval medzi nimi. Ako stimuly môžeme použiť text, načítané obrázky

vo formáte BMP alebo zvukové nahrávky. Okrem toho existuje mnoho ďalších parametrov určujúcich vlastnosti zvukových stimulov (hlasitosť...) alebo okna aplikácie.

My sme túto aplikáciu používali hlavne na zistenie príznakov (features), ktoré sme potom využívali v aplikácii *Cursor Task*. Upravenú aplikáciu Stimulus Presentation sme taktiež používali na vytvorenie trénovacej množiny pre vytvoreni LDA klasifikátora (pozri časť 4.3)

P300 Speller Task

Táto aplikácia je implementáciou P300 paradigmy popísanej v časti 1.3.2. Užívateľovi sa zobrazí štandardne matica 6×6 znakov. Postupne sa zvýrazňujú jednotlivé stĺpce a riadky. Úlohou užívateľa je vykonávať nejakú úlohu, keď je želaný znak (ten, ktorý chce napísať) zvýraznený. Aplikácia P300 Speller Task v spolupráci s filtrom *P3Temporal Filter* rozhoduje, aký znak chcel užívateľ napísať.

Pomocou parametrov sa nastavujú rôzne vlastnosti experimentu ako sú vlastnosti okna, matica znakov, rýchlosť zvýrazňovania riadkov a stĺpcov, alebo počet znakov, ktoré treba počas experimentu napísať.

2.4 Možnosti rozšírenia BCI2000

Ako sme spomínali v úvode, BCI2000 je otvorený projekt a teda jeho zdrojové kódy sú voľne dostupné. Aplikáciu je možné ľubovoľne upravovať a prispôsobovať. Taktiež sa dajú naprogramovať vlastné moduly alebo filtre. Vývojári nám pripravili nástroje, vďaka ktorým môžeme BCI2000 považovať za framework, v ktorom sa dajú ľahko implementovať BCI aplikácie. Práve pre túto vlastnosť sme si ho vybrali v našej práci.

Na webovej stránke projektu alebo v knihe Schalk et al. (2004) je presne popísaný postup, ako získať zdrojové kódy a ako ich skompilovať. My sme na prístup k zdrojovým kódom používali SVN klienta TortoiseSVN a ako vývojové prostredie Microsoft Visual Studio 2008. Po získaní súborov vznikne podobná adresárová štruktúra ako tá, ktorú sme spomínali v časti 2.1. Pribudnú iba dva nové priečinky:

- src - zdrojové kódy ku všetkým častiam BCI2000,
- build - súbory a nástroje potrebné na vytvorenie projektu pre jednotlivé podporované vývojové prostredia.

Zdrojové kódy sú písané v programovacom jazyku C++ a sú nezávislé od vývojového prostredia. Keď si užívateľ vyberie prostredie, v ktorom chce vyvýjať, tak z priečinku build spustí príslušný súbor (v našom prípade to bol Make VS2008 Project Files.bat), ktorý vytvorí projekt pre vybraté prostredie. Potom ho stačí iba načítať v prostredí a môže sa začať samotný vývoj.

2.4.1 Implementácia vlastného modulu a filtra

Jedným z cieľov našej práce bolo implementovať v BCI2000 klasifikátor LDA (pozri časť 3.1) Na to sme potrebovali naprogramovať filter, ktorý bude robiť samotnú klasifikáciu a následne ho spolu s ďalšími potrebnými vložiť do nového modulu typu Signal Processing. Keďže to je jedna z hlavných možností, ako môžeme BCI2000 upraviť, tak nám vývojári na to pripravili nástroje, ktoré ich pre nás vytvoria podľa šablóny. V priečinku build (po vytvorení projektu) sa nachádzajú nástroje *NewBCI2000Filter.exe* a *NewBCI2000Module.exe*. Sú to aplikácie, ktoré si od vás vypýtajú základné údaje o novom fitri/moduli ako jeho meno, typ a priečinok, kde má byť vytvorený. Štandardne sa vytvárajú do priečinku src/custom. Po tomto kroku treba opäť nechať vytvoriť projekt (priečinok buid) aby sa novo vytvorený filter alebo modul objavil v projekte.

Po otvorení projektu v prostredí Microsoft Visual Studio 2008 sa zobrazia jednotlivé moduly a aplikácie. Pri programovaní nového modulu (napr. typu Signal Processing) je najdôležitejší súbor PipeDefinition.cpp. V ňom je napísané, aké filtre obsahuje a aké je ich poradie.

Každý fiter je štandardná trieda obsahujúca konštruktor, deštruktor, definíciu parametrov a množinu funkcií. Na začiatku filtra je popis parametrov, ktoré bude používať. Tie môžu byť rôzneho typu (čísla, reťazce, matice, zoznamy...). Najdôležitejšie sú nasledovné funkcie, ktoré zabezpečujú jeho funkcionalitu:

- Preflight - spustí sa, keď užívateľ stlačí tlačidlo Set Config. Jej úlohou je skontrolovať, či všetky parametre sú v správnej forme a v správnom rozsahu. Taktiež zisťuje, či sú všetky parametre a stavy, ktoré sa vo filtri používajú dostupné (viditeľné) a nastavuje vlastnosti výstupu (hlavne jeho veľkosť),

- Initialize - spustí sa, keď funkcia Preflight skončila bez problémov a môže sa pokračovať ďalej. Vtedy sa už nedajú meniť vlastnosti signálu. V tejto časti sa inicializujú všetky lokálne dátové štruktúry potrebné na prácu filtra,

- Process - vykonáva samotnú funkcionalitu filtra. Tu sa definuje ako transformuje filter svoje vstupy na výstupy. Vykonáva sa vždy, keď zosilňovač pošle blok dát (vzorkovacia frekvencia/veľkosť bloku - krát za sekundu),
- StartRun - spustí sa pred začiatkom každej sady pokusov. Slúži na nastavenie hodnôt pre nasledovnú sadu, ako je napríklad skopírovanie parametrov do lokálnych štruktúr, otvorenie súborov, ktoré filter používa atď,
- StopRun - spúšťa sa po skončení sady pokusov. Slúži na uvoľnenie prostriedkov v nej použitých alebo napr. na uloženie vypočítaných hodnôt do parametrov.

Po implementácii popísaných funkcií filtra ho stačí umiestniť do vhodného modulu, ktorý v kombinácii s ostatnými filtrami (napr. *Normalizer*, *LP Filter*, *Spatial Filter*...) vykonáva želanú funkcionalitu. Pre spustenie experimentu s novým modulom si treba napísať spúštací skript, podobný, ako tie v priečinku batch, ktorý spustí vhodnú kombináciu Source Modulu, Data Processing Modulu a koncovej aplikácie.

Kapitola 3

Použité metódy

Jedným z cieľov našej práce bolo naimplementovať iný druh klasifikátora ako bol pôvodne poskytnutý aplikáciou BCI2000, použiť ho počas online experimentu a vyhodnotiť jeho úspešnosť. Štandardná inštalácia BCI2000 poskytuje filter Linear Classifier, ktorý na výstup vracia iba dopredu definovanú lineárnu kombináciu svojich vstupov (pozri časť 2.3.2). Samotnú klasifikáciu však nevie vykonávať. Tú robí až filter Normalizer tým, že svoje vstupy normalizuje na nulový priemer a jednotkový rozptyl. Klasifikácia do dvoch tried potom prebieha na základe toho, či je výstup kladný alebo záporný.

Daný filter sa dokáže počas behu programu adaptovať. Pri adaptácii však využíva informáciu o triede, do ktorej sa má klasifikovať. Takýto typ učenia sa volá *učenie s učiteľom*. Dosahuje veľmi dobré výsledky, ale v reálnom živote nemáme vždy informáciu o tom, čo chcel užívateľ vykonať. Predstavme si ovládanie robotickej ruky. Tá sa musí stále adaptovať a tým reagovať na zmeny v mozgovej činnosti subjektu, ale on jej neposkytuje informáciu o tom, aký pohyb chcel vykonať. Musí to zvládať aj bez toho. Preto sme v našej práci vyskúšali klasifikátor, ktorý sa dokáže adaptovať metódou *učenie bez učiteľa*. Existuje viac možností ako to spraviť, my sme sa zamerali na adaptácie v rámci LDA (Linear Discriminant Analysis).

3.1 Lineárna diskriminačná analýza

Lineárna diskriminačná analýza (LDA) je jenoduchá metóda používaná v štatistike alebo strojovom učení. Jej úlohou je hľadať takú lineárnu kombináciu príznakov, ktorá najlepšie charakterizuje, alebo oddeľuje prvky dvoch tried. Takáto kombinácia sa môže použiť na redukciu dimenzionality dát, alebo ako lineárny klasifikátor.

My sme túto metódu požívali ako klasifikátor kvôli nasledovným dôvodom:

- dokáže sa adaptovať metódou bez učiteľa,
- jej implementácia je výpočtovo nenáročná a preto je vhodná pre online experimenty so spätnou väzbou,
- je vhodná na klasifikáciu do dvoch tried.

Úlohou klasifikátora je rozhodovať, do ktorej z tried patrí vstup. Počas našich experimentov sme potrebovali iba klasifikáciu do dvoch tried (či má ísť loptička hore, alebo dole). Pred tým, ako chceme klasifikátor používať, tak ho najprv musíme naučiť na trénovacích dátach (pre detaily nášho experimentu pozri časť 4.3). Po naučení je LDA klasifikátor definovaný ako nadrovina (v 2D je to priamka, v 3D rovina), ktorá oddeľuje dané dve triedy prvkov. Takáto nadrovina je určená normálovým vektorom $\boldsymbol{w}$ určujúcim jej orientáciu a číslom b, ktoré keď predelíme zložkami vektora $\boldsymbol{w}$ tak dostaneme body, v ktorých nadrovina pretína osi.

Na klasifikáciu potrebujeme ešte priemer hodnôt pre obe triedy (vektory $\boldsymbol{\mu}_1$ a $\boldsymbol{\mu}_2$) a kovariančnú maticu $\boldsymbol{C}$. Pre náhodný vstupný vektor $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ je kovariančná matica štvorcová symetrická matica s rozmermi $p \times p$, ktorá obsahuje na priesečníku i-teho riadku a j-teho stĺpca kovarianciu prvkov x_i a x_j. Takáto matica je pre náhodnú premennú $\boldsymbol{x}$ definovaná nasledovne:

$$\boldsymbol{C}_x = \frac{1}{N}\sum_{t=1}^{N}(\boldsymbol{x}(t) - \boldsymbol{\mu}_x)(\boldsymbol{x}(t) - \boldsymbol{\mu}_x)^\top \tag{3.1}$$

pričom $\boldsymbol{\mu}_x$ je priemer hodnôt všetkých prvkov a N je ich celkový počet. Kovariančnú maticu sme získavali zo všetkých trénovacích dát v aplikácii Matlab pomocou funkcie `cov(x)`, ktorá implementuje rovnicu 3.1.

LDA klasifikátor je potom definovaný nasledovnými rovnicami:

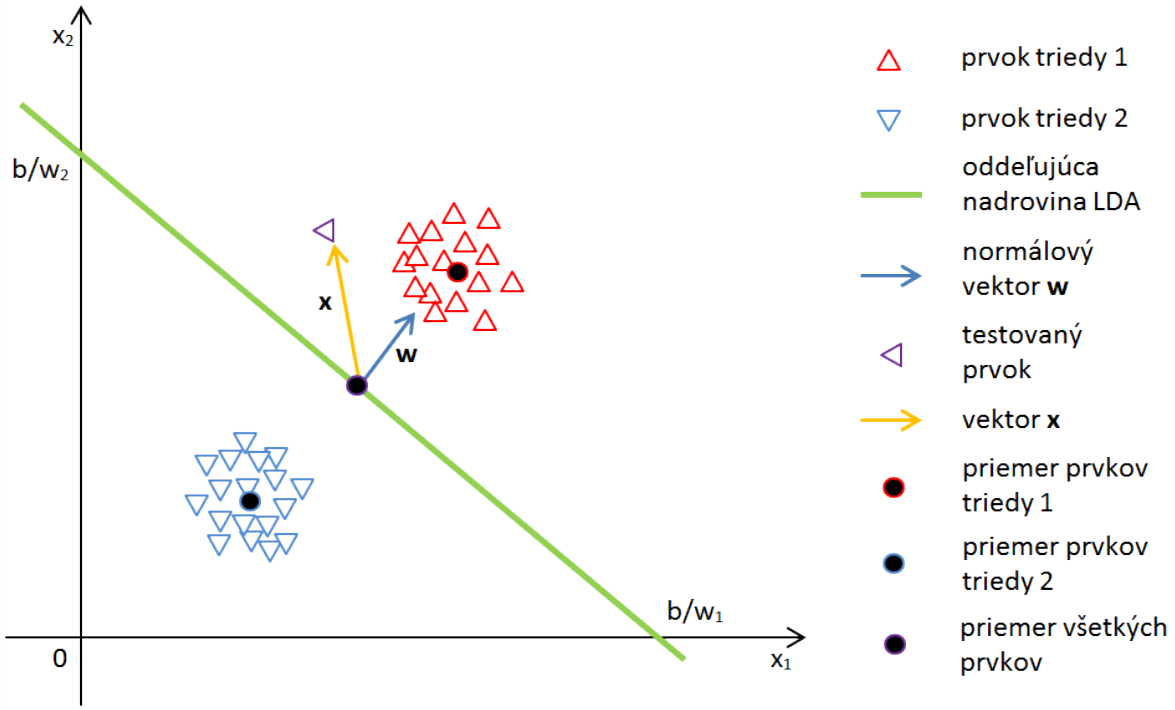
$$D(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{3.2}$$

$$\boldsymbol{w} = \boldsymbol{C}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \tag{3.3}$$

$$b = -\boldsymbol{w}^\top \boldsymbol{\mu} \tag{3.4}$$

$$\boldsymbol{\mu} = \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) \tag{3.5}$$

kde $D(\boldsymbol{x})$ určuje vzdialenosť vstupného vektora $\boldsymbol{x}$ od oddeľovacej nadroviny a operátor $^\top$ označuje transponovanie. Znamienko $D(\boldsymbol{x})$ určuje, na ktorej strane nadroviny sa vstup nachádza a tým určuje triedu, do ktorej patrí.



Obr. 3.1: Vizualizácia LDA pre 2D.

LDA vysvetlíme na 2D príklade zobrazenom na obr. 3.1. V našom 2D priestore sa nachádzajú prvky dvoch tried (červené a modré trojuholníky). Oddeľujúcu nadrovinu natrénovaného LDA klasifikátora sme znázornili zelenou priamkou. Tá je jednoznačne určená normálovým vektorom $\boldsymbol{w}$ a číslom b, ktoré keď predelíme jednotlivými zložkami vektora $\boldsymbol{w}$ tak získame body, v ktorých priamka pretína osi x_1 a x_2. Ak chceme klasifikovať nejaký prvok (fialový trojuholník), tak rovnica 3.2 reprezentuje skalárny súčin vektorov $\boldsymbol{w}$ a $\boldsymbol{x}$ (začína v rovnakom bode ako vektor $\boldsymbol{w}$, ktorým je celkový primer všetkých prvkov $\boldsymbol{\mu}$, a končí v klasifikovanom prvku). Ak je skalárny súčin kladný, tak je uhol daných vektorov menší ako 90° a tým pádom testovaný prvok patrí na tú istú stranu priamky ako vektor $\boldsymbol{w}$ čiže do triedy 1. V prípade zápornej hodnoty je uhol väčší ako 90° a prvok sa nachádza na opačnej strane, takže patrí do triedy 2.

3.2 Typy adaptácie LDA

Úplne ideálne by bolo, keby stačilo klasifikátor raz naučiť a potom by stále podával rovnako dobré výsledky. Problém je v tom, že EEG signál nie je stacionárny a teda jeho statické vlastnosti sa v čase menia. Okrem neho sa mení aj mozgová činnosť subjektu, ktorý vykonáva nejaký experiment. Natrénovaný klasifikátor, ktorý jeden deň podával

výborné výsledky, o týždeň už nemusí byť taký dobrý. Na priebeh signálu vplýva mnoho faktorov - presné umiestnenie elektród, čas, kedy sa experiment vykonáva, únava a psychické rozpoloženie subjektu a iné faktory. Ak nechceme pred každým meraním učiť klasifikátor nanovo (čo môže byť časovo náročné), tak musí byť schopný sa prispôsobiť (adaptovať).

Existuje viacero spôsobov ako adaptovať LDA klasifikátor. V našej práci sme implementovali a následne skúmali tri typy. Kvôli porovnaniu s pôvodným klasifikátorom v BCI2000 sme zvolili jednu metódu typu *učenie s učiteľom* a nasledovne dve typu *učenie bez učiteľa*. V našich experimentoch dochádzalo k adaptácii počas spätnej väzby 32-krát za sekundu.

3.2.1 Adaptácia LDA s učiteľom

Pri tomto type adaptácie dochádza k úprave priemerov hodnôt pre každú z tried zvlášť (vektory $\boldsymbol{\mu}_1$ a $\boldsymbol{\mu}_2$) a taktiež k zmene kovariančnej matice $\boldsymbol{C}$. Tým sa mení sklon oddeľujúcej nadroviny aj jej posun oproti začiatku súradnicovej sústavy. V práci budeme takýto typ adaptácie označovať *Supervised*. Priemery hodnôt sa menia rekurzívne pomocou rovnice vyhladzovacieho spriemerňovania

$$\boldsymbol{\mu}_i(t) = (1-\alpha)\boldsymbol{\mu}_i(t-1) + \alpha\boldsymbol{x}(t) \tag{3.6}$$

kde α je konštanta určujúca rýchlosť učenia, t udáva index vzorky v čase, $\boldsymbol{x}$ je vstup a i je index priemeru, ktorý nadobúda hodnotu 1 alebo 2 podľa toho, aký cieľ bol zobrazený (hore alebo dole). Hodnota $\boldsymbol{\mu}(0)$ je určená z trénovacích dát. Rýchlosť učenia bola na začiatku určená podľa článku Vidaurre et al. (2010) na α=0.05. Táto metóda je typu *učenie s učiteľom* preto, lebo sa mení iba jeden z priemerov podľa informácie o triede, do ktorej sa má klasifikovať.

LDA závisí od inverznej kovariančnej matice $\boldsymbol{C}^{-1}$ avšak počítanie inverznej matice je výpočtovo náročná operácia. Našťastie však existuje rekurzívna rovnica, v ktorej sa nachádzajú iba základné operácie ako násobenie matice maticou, alebo násobenie matice číslom (Vidaurre et al., 2010):

$$\boldsymbol{C}(t)^{-1} = \frac{1}{(1-\beta)}\left(\boldsymbol{C}(t-1)^{-1} - \frac{\boldsymbol{v}(t)\boldsymbol{v}(t)^\top}{\frac{1-\beta}{\beta} + \boldsymbol{x}(t)^\top\boldsymbol{v}(t)}\right) \tag{3.7}$$

$$\boldsymbol{v}(t) = \boldsymbol{C}(t-1)^{-1}\boldsymbol{x}(t) \tag{3.8}$$

kde β určuje rýchlosť učenia, $\boldsymbol{x}$ je vstup a $\boldsymbol{C}(t-1)^{-1}$ je hodnota inverznej kovariančnej matice v predchádzajúcom časovom kroku. Inverzná matica sa teda počíta iba pri prvotnom učení klasifikátora z trénovacích dát.

3.2.2 Adaptácia LDA bez učiteľa I

Tento typ adaptácie je založený na zmene celkového priemeru hodnôt $\boldsymbol{\mu}$ pričom vzdialenosť medzi priemermi oboch tried $(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ a kovariančná matica ostávajú od naučenia na trénovacích dátach konštantné. Úprava priemeru $\boldsymbol{\mu}$ prebieha podľa rovnice 3.6 rovnako ako pri Supervised adaptácii. Pri takejto adaptácii sa mení iba posun oddeľovacej nadroviny a nie jej sklon. Budeme ju označovať ako *Unsupervised I.*

3.2.3 Adaptácia LDA bez učiteľa II

Pri poslednom type adaptácie sa mení celkový priemer $\boldsymbol{\mu}$ rovnako ako pri Unsupervised I a k tomu sa pridá zmena inverznej kovariančnej matice $\boldsymbol{C}^{-1}$ podľa rovnice 3.7. Tým pádom sa mení sklon oddeľujúcej nadroviny aj jej posun a to všetko bez informácie o triede, do ktorej sa má klasifikovať. Budeme tento spôsob preto označovať ako *Unsupervised II.*

Kapitola 4

Experimenty

Jedným z cieľov našej práce bolo vyskúšať experimenty so spätnou väzbou a porovnať úspešnosť rôznych variant LDA klasifikátora. Na to však bolo treba najprv zistiť, či sa pri subjektoch prejavuje desynchronizácia μ-rytmu, vybrať najvhodnejšie dve činnosti, ktoré bude vykonávať a natrénovať samotný klasifikátor na trénovacích dátach.

Keďže desynchronizácia μ-rytmu a je najviac pozorovateľné nad motorickou časťou kôry (pozri časť 1.3.2) tak sme počas našich experimentov merali nasledovné elektródy: T7, C3, F3, Fz, Cz, F4, C4 a T8. Počas našej práce sme vykonali viac ako 60 meraní, pričom každé trvalo v priemere tri hodiny.

4.1 Kontrola prítomnosti desynchronizácie μ-rytmu

Naše experimenty začínali zistiťovaním, či sa pri meraných subjektoch prejavuje ERD, ktorú sme popisovali v časti 1.3.2. To bolo hlavným predpokladom úspešných experimentov so spätnou väzbou. Pomocou aplikácie g.Recorder sme namerali jednoduchý pokus, obsahujúci štyri úlohy, pričom každá trvala 30 sekúnd.

Na začiatku sme zvolili nejaký druh pohybu, väčšinou ťukanie ukazovákom. Prvou úlohou subjektu bolo vybraný pohyb vykonávať. Potom sa do experimentu zapojil druhý človek, ktorý začal robiť to isté a úlohou meraného subjektu bolo pohyb pozorovať. Potom si mal meraný človek daný úkon iba predstavovať. Celý experiment končil relaxom. Opakovali sme ho dvakrát pre každú ruku. Takto sme otestovali dokopy 10 rôznych subjektov.

Namerané dáta sme použili ako vstup do nami napísaného skriptu v Matlabe (pozri časť 1.2.2), ktorý počítal spektrálnu analýzu. Pri tomto experimente obsahoval

výsledný graf štyri krivky, jednu pre každú úlohu. Z teórie vieme, že pri pohybe človeka dochádza k poklesu napätia nad motorickou časťou mozgovej kôry vo frekvenčnom pásme μ-rytmu. Po dokončení pohybu a následnej relaxácii dochádza k opätovnému zvýšeniu napätia na pôvodnú hodnotu. V ideálnom prípade sme teda očakávali, že krivky reprezentujúce vykonávaný, pozorovaný a predstavovaný pohyb mali podobný priebeh a v oblasti μ-rytmu dosahovali nízke hodnoty. V prípade relaxu sme očakávali, že v oblasti μ-rytmu bude dosahovať signifikantne väčšiu hodnotu ako pri predchádzajúcich úlohách. Štandardne sme merali 8 kanálov a najväčšie rozdiely sme očakávali na centrálnych elektródach (C3, C4 a Cz).

4.1.1 Výsledky

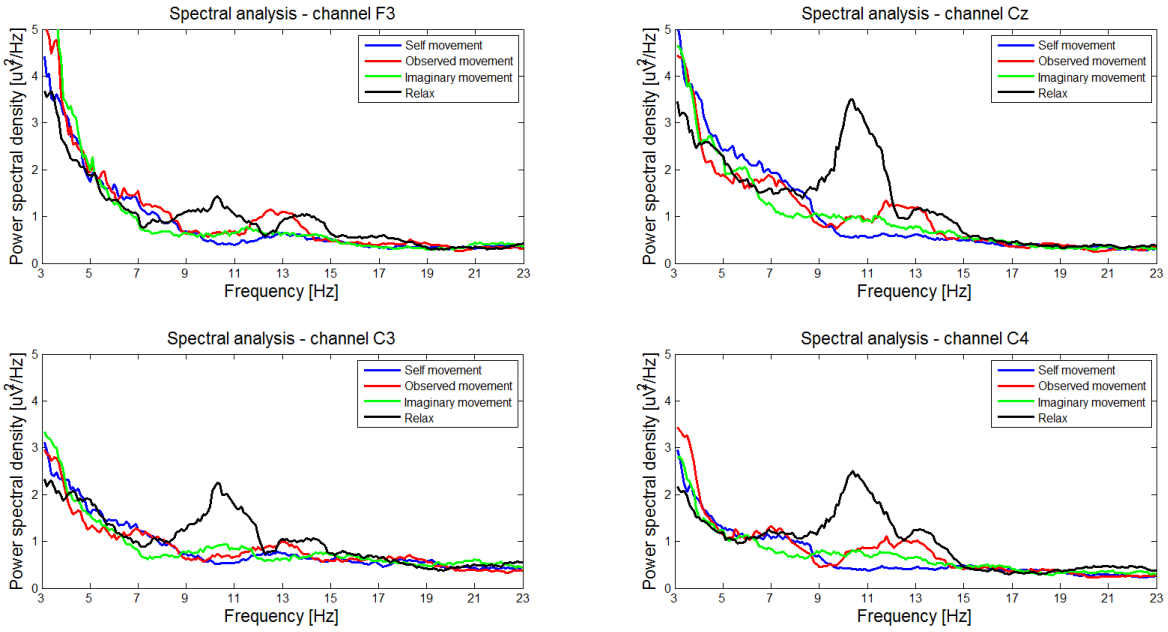
Tento experiment sme skúšali ako náš prvý ešte v čase, keď sme sa učili správne merať subjekty. Preto nie všetky výsledky vyšli podľa očakávania. Niektoré subjekty zo začiatku vôbec nevykazovali rozdiel medzi relaxom a pohybom. U iných sa prejavoval rozdiel medzi relaxom a vykonávaným pohybom, ale predstavovaný pohyb nemal priebeh, ako sme chceli. Pri niektorých subjektoch sa zmena prejavovala pri jednej ruke signifikantnejšie ako pri druhej. Pri jednom človeku sa stalo, že predstavovaný pohyb bol podobný ako vykonávaný, ale pozorovaný bol skôr podobný relaxu.

Keď sme však postupom času začali pripravovať experimenty so spätnou väzbou (popísané v časti 4.4), tak merané subjekty si po chvíli trénovania osvojili predstavovaný pohyb a jeho krivka mala potom podobný priebeh ako ten vykonávaný. Na obr. 4.1 je zobrazená ukážka výsledkov spektrálnej analýzy tohto experimentu pre vybraný subjekt, ktorý dosahoval očakávané výsledky.

4.2 Výber najvhodnejších podmienok

Keď sme už zistili, že sa pre daný subjekt objavuje rozdiel medzi predstavovaným pohybom a relaxom, tak nasledovala druhá časť a tou bolo zistenie, ktoré dve úlohy sú pre daného človeka najvhodnejšie. Rozličné predstavované pohyby vyvolávali rozlične veľkú desynchronizáciu a naopak, pri relaxovanom stave sme tiež dokázali dosahovať rozličné výsledky.

Prvým krokom bolo zistiť, aký predstavovaný pohyb sa subjektu najľahšie predstavuje a prejavuje sa pri ňom najväčšia desynchronizácia μ-rytmu. Prvým skúšaným pohybom bolo ťukanie ukazovákom. Okrem neho sme skúšali zovretie ruky v päsť. Po-

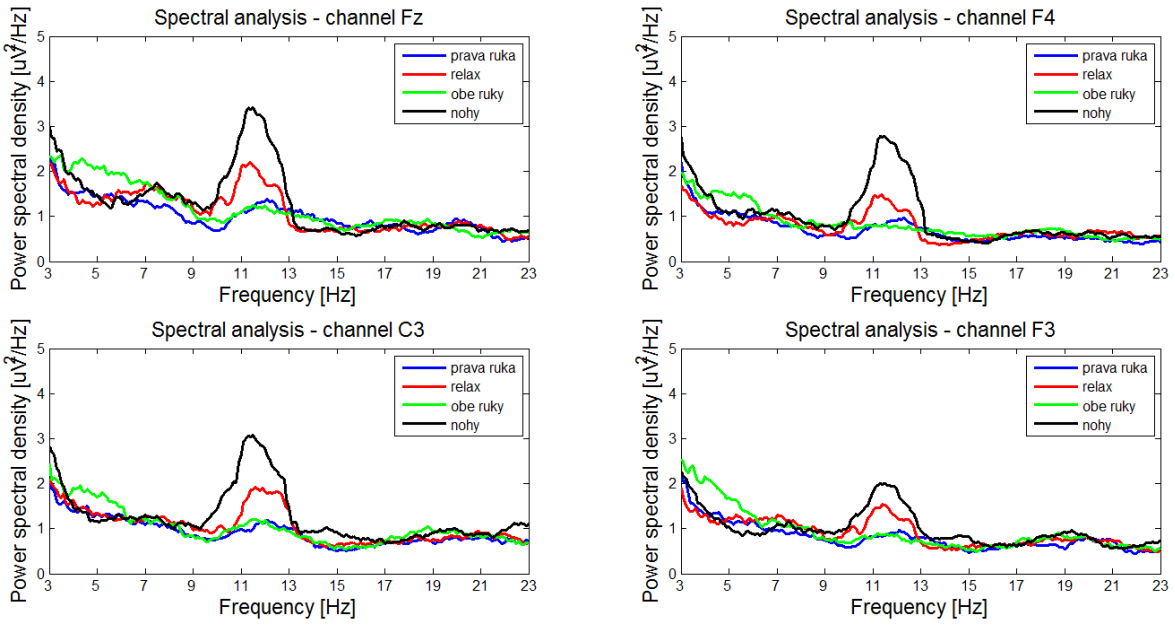


Obr. 4.1: Príklad výsledkov spektrálnej analýzy na vybraných elektródach pre pokus *Určovanie prítomnosti desynchronizácie μ-rytmu.*

mocou aplikácie g.Recorder sme namerali jednotlivé predstavované pohyby a pomocou spektrálnej analýzy vyhodnocovali. Keď sme už vybrali najlepší pohyb, tak sme ešte skúšali, ktorá ruka je dominantnejšia, či ľavá, pravá, alebo ich kombinácia (subjekt vtedy vykonával vybratý predstavovaný pohyb oboma rukami).

Niektoré subjekty mali problém zrelaxovať, keď im bolo povedané: „relaxuj a nepredstavuj si žiadny pohyb!“. Osoby boli vtedy príliš sústredené na to, aby dobre relaxovali, až sa im to vôbec nedarilo. Niektoré subjekty sa však časom naučili navodiť si stav relaxu. Veľkú pomoc v tomto probléme priniesla RNDr. Barbora Cimrová, ktorá keď bola meraná, tak vyhlásila, že si ako relax predstavuje plutvový pohyb nohami. Podľa výsledkov na nej a aj na iných subjektoch sme zistili, že predstava nejakej situácie, ktorú subjekt považuje za relaxačnú (napr. kúpem sa pri mori, mám plutvy a plávem) dokáže vyvolať vyvolať vyššie napätie nad motorickou časťou mozgovej kôry vo frekvenčnom pásme μ-rytmu ako samotný povel „relaxuj“. Tento poznatok môžete vidieť aj na obr. 4.2, kde boli merané štyri rôzne činnosti: predstavovaný pohyb pravou rukou, relax, predstavovaný pohyb oboma rukami a predstavovaný plutvový pohyb nohami. Krivka pohybu s nohami je v obasti μ-rytmu vždy vyššie ako krivka relaxu. Krivky pohybu pravou rukou a oboma rukami majú približne rovnaký priebeh.

Experimenty so spätnou väzbou sme merali iba pre 4 subjekty a pre nich najvhodnejšou kombináciou stavov boli: ťukanie ľavým ukazovákom vs. plutvový pohyb



Obr. 4.2: Príklad výsledkov spektrálnej analýzy na vybraných elektródach pre experiment zisťovania najvhodnejších činností pre experimenty so spätnou väzbou.

nohami, zovieranie pravej ruky v päsť vs. plutvový pohyb nohami, ťukanie pravým ukazovákom vs. relax (nie pohyb nohami, pre tento subjekt to nebolo lepšie) a ťukanie oboma ukazovákmi vs. plutvový pohyb nohami.

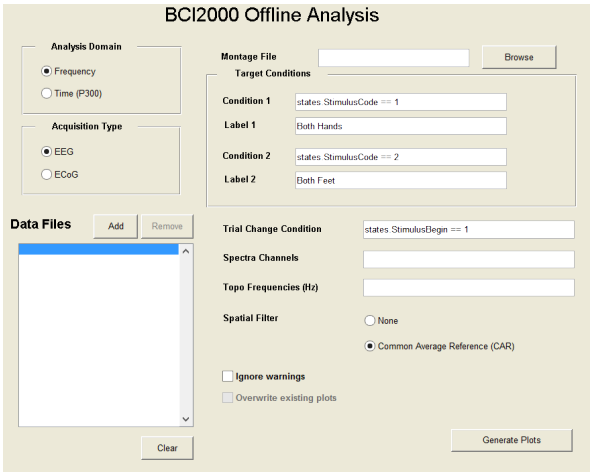
4.3 Získavanie príznakov a tvorba LDA

Ak sme už vybrali dve najvhodnejšie činnosti, ktoré bude subjekt vykonávať, tak prišiel čas na vybratie vhodných príznakov a natrénovanie LDA klasifikátora, ktorý sa používal neskôr pri experimentoch so spätnou väzbou (pozri časť 4.4).

Na to sme však potrebovali nejaké trénovacie dáta. Použili sme aplikáciu *Stimulus presentation* (pozri časť 2.3.3), pomocou ktorej sme subjektu prezentovali dve šípky, jednu smerujúcu dole a druhú hore. Pri šípke dole mal vykonávať vybraný predstavovaný pohyb a pri šípke hore relaxovať. Každý stimul bol zobrazený na 4 sekundy a dokopy ich bolo prezentovaných 20 pre každý druh počas jednej sady pokusov. Na vytvorenie relevantnej trénovacej množiny sme merali dva až tri sady, takže dokopy sme mali 40–60 vzoriek pre obidve činnosti.

4.3.1 Získavanie príznakov

Získavanie príznakov (feature extraction) prebiehalo offline pomocou aplikácie *OfflineAnalysis*, ktorá bola súčasťou BCI2000. Bola naprogramovaná v Matlabe a obsahovala jednoduché grafické rozhranie zobrazené na obr. 4.3.

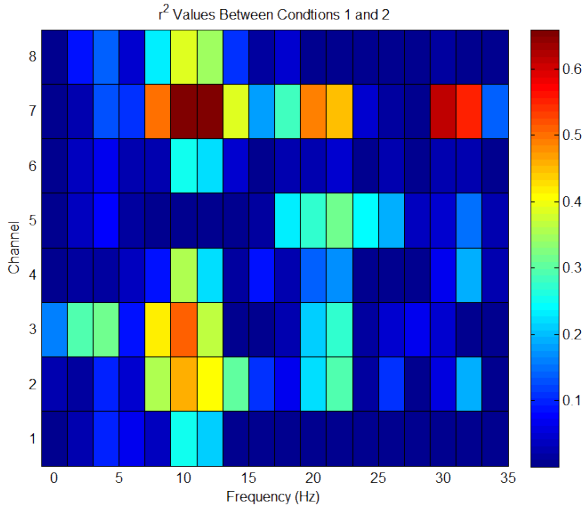


Obr. 4.3: Užívateľské rozhranie aplikácie OfflineAnalysis z BCI2000 používanej na získavanie príznakov pre jednotlivé subjekty.

Do aplikácie sa načítali trénovacie dáta a vybrali sa dve činnosti, ktoré sa mali porovnávať. My sme merali iba dva stimuly (šípka hore a dole). Dal sa ešte nastaviť typ priestorového filtra (spatial filter, pozri časť 2.3.2) a označenia vybraných dvoch činností, aby sa v grafoch dali ľahšie rozoznať. Úlohou aplikácie bolo zistiť, že na ktorých elektródach a na akých frekvenciách na nich vzniká pre dané dve činnosti najväčší rozdiel v napätiach získaných pomocou spektrálnej analýzy. Výsledok pre vybraný subjekt je zobrayený na obr. 4.4.

Vo výsledku sa na osi x nachádzajú frekvencie a na osi y indexy meraných kanálov. Farba blízka modrej vyjadruje malý rozdiel medzi činnosťami a farba blízka hnedej signalizuje veľký rozdiel na danej frekvencii pre danú elektródu. Aplikácia umožňuje vybrať si niektoré kanály a následne zobraziť výsledok spektrálnej analýzy, na základe ktorej vznikol výsledný obrázok. Z takýchto grafov sa sá potom zistiť, či sa priebeh kriviek podobá tomu, čo sme videli v častiach 4.1 a 4.2 Z obr. 4.4 je zrejmé, že najvhodnejšie príznaky sú: 9Hz a 19Hz na elektróde 7 (C4), 9Hz na elektróde 3 (F3) a 9Hz na elektróde 2 (C3).

Existujú aj iné metódy ako získavať príznaky. Aplikovaním metódy Common Spatial Patterns na trénovaciu množinu ich vieme získať automaticky bez nutnosti offline



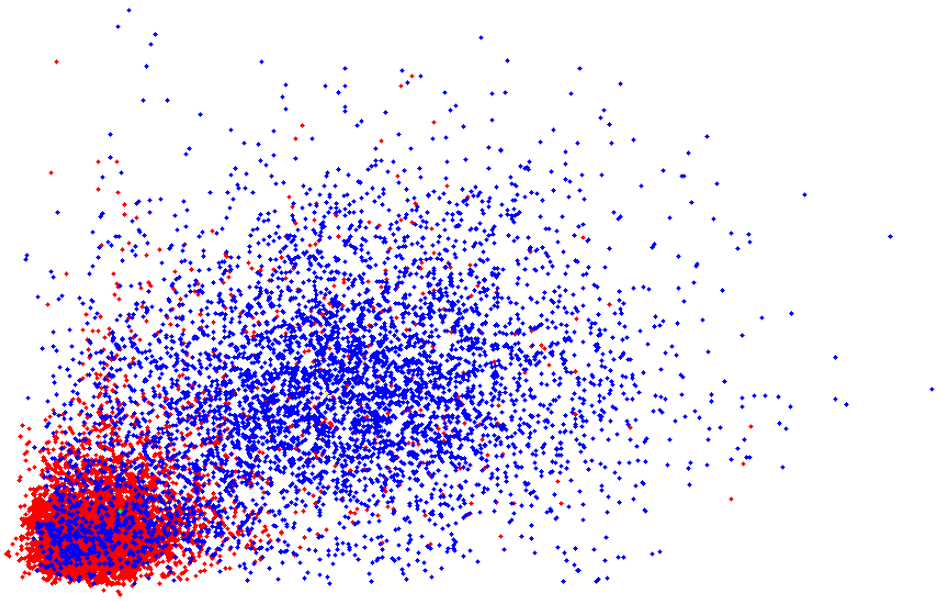
Obr. 4.4: Výsledok offline analýzy na získanie príznakov pre vybraný subjekt.

analýzy a prítomnosti osoby, ktorá ich pri našich pokusoch musela manuálne vyberať (Blankertz et al., 2008). Metóda hľadá takú projekciu dát, ktorá maximalizuje varianciu jednej činnosti a minimalizuje varianciu druhej. Použitie takéhoto prístupu zrýchľuje fázu prípravy na online pokusy so spätnou väzbou a malo by zabezpečiť lepšiu rozlíšiteľnosť vstupných dát a tým presnejšiu klasifikáciu (Hoffmann, 2012). V našej práci sme túto metódu z časových dôvodov neimplementovali.

4.3.2 Tvorba LDA klasifikátora

Posledným krokom pred experimentami so spätnou väzbou bolo vytvorenie LDA klasifikátora z trénovacej množiny. Na to sme si napísali skript v Matlabe, ktorý načítal trénovacie dáta a následne na priestore príznakov (príznaky boli vstupom skriptu) vypočítal kovariančnú maticu a priemery pre prvky oboch tried. Všetky vypočítané údaje potom uložil do súboru, ktorý sme neskôr načítavali v BCI2000 počas experimentov so spätnou väzbou.

Pomocou metódy *k-fold Cross validation* skript ešte vypočítal úspešnosť vytvoreného LDA klasifikátora. Pre naše štyri subjekty boli výsledky nasledovné (zoradené vzostupne): 65,9%, 68,3%, 69,2% a 85,2%. Posledou úlohou programu bolo vykreslenie trénovacích dát z dôvodu vizuálnej kontroly rozlíšiteľnosti prvkov daných dvoch tried. Keďže priestor príznakov mohol byť mnohorozmerný, tak kvôli vizualizácii bolo treba vybrať dva také rozmery, pre ktoré boli prvky najviac rozlíšiteľné. Graf dvoch vybratých rozmerov pre náš najúspešnejší subjekt môžete vidieť na obr. 4.5. Modré prvky reprezentujú stav relaxu a červené predstavovaný pohyb.



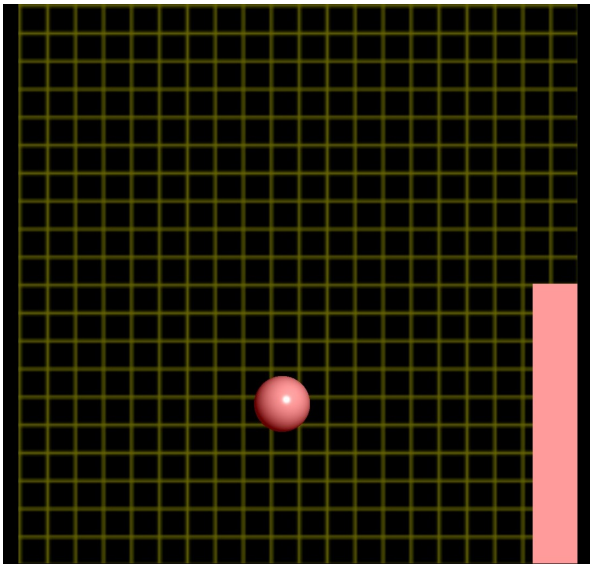
Obr. 4.5: Príklad zobrazenia trénovacích dát v priestore príznakov na vybraných dvoch rozmeroch zo siedmich. Úspešnosť LDA klasifikátora natrénovaného na týchto dátach bola 85,2%.

4.4 Online experimenty so spätnou väzbou

Po výbere správnych pohybov, ktoré sú pre subjekt najľahšie rozlíšiteľné a výbere vhodných príznakov (popísané v častiach 4.2 a 4.3) nasledovali online experimenty so spätnou väzbou.

4.4.1 Zadanie experimentu

Úlohou subjektu bolo pomocou dvoch imaginárnych pohybov ovládať pohyb kurzora po obrazovke a snažiť sa trafiť zobrazený cieľ. Na to sme používali aplikáciu CursorTask z programu BCI2000 (pozri časť 2.3.3). Kurzor bol reprezentovaný guľôčkou a cieľ farebne odlíšeným obdĺžnikom, ktorý sa objavil na pravom okraji obrazovky, buď dole alebo hore. Potom sa na ľavom okraji obrazovky zjavil kurzor a začal sa pohybovať konštantnou rýchlosťou smerom doprava. Jedna cesta kurzora z ľavého okraja po pravý sa označoval ako pokus. Na obr. 4.6 môžete vidieť ako vyzerala obrazovka aplikácie, ktorú videl subjekt.



Obr. 4.6: Ukážka obrazovky počas experimentu so spätnou väzbou, aplikácia CursorTask.

4.4.2 Použité nastavenia

V našej práci sme merali štyri subjekty, jednu ženu a troch mužov vo veku od 23 do 45 rokov. Každý bol meraný trikrát a rozostupy medzi jednotlivými sedeniami boli v intervale jedného až štyroch týždňov. Jedno sedenie pozostávalo štandardne z 10 až 15 sád pokusov. Každá sada pokusov obsahovala desať pokusov pre každý z cieľov (hore a dole), čiže dokopy 20. V jednej sade bol výber polohy cieľa náhodný, pričom sa dodržoval ich počet. Počas každého sedenia sa merali sady pokusov s pôvodne poskytovaným klasifikátorom aj s klasifikátorom LDA, menili sme však typy jeho adaptácie (pozri časť 3.2). Dokopy sme merali nasledovné varianty ovládania kurzoru, ktoré môžeme nazývať klasifikátormi:

- Standard - klasifikátor ponúkaný štandardnou inštaláciou aplikácie BCI2000. Adaptácia prebihala na strane filtra Normalizer (časť 2.3.2) metódou *učenie s učiteľom*,
- WithoutAdaptation - offline natrénovaný LDA klasifikátor bez adaptácie,
- Supervised - LDA adaptované metódou *učenie s učiteľom*,
- Unsupervised I - LDA adaptované metódou *učenie bez učiteľa* typu I (časť 3.2.2),
- Unsupervised II - LDA adaptované inou metódou *učenie bez učiteľa* (časť 3.2.2).

Časovanie jednej sady pokusov bolo nasledovné. Najprv sa subjektu na 5 sekúnd zjavila výzva na sústredenie sa. Potom začali jednotlivé pokusy, ktoré boli oddelené od seba sekundovou pauzou, počas ktorej sa zobrazovala prázdna obrazovka. Jeden pokus sa skladal z troch častí: najprv sa zobrazil na 2 sekundy cieľ, potom sa zjavil kurzor a začal sa pohybovať (tejto fáze hovoríme spätná väzba) a po trafení alebo netrafení cieľa sa na jednu sekundu zobrazil výsledok pokusu. Ak subjekt trafil cieľ, tak cieľ zmenil farbu. Dĺžka spätnej väzby bola nastavená na 4 sekundy (čas, za ktorý by kurzor prešiel celú obrazovku) ale keďže skončila po zasiahnutí cieľa kurzorom a kurzor má svoju šírku, tak nestihol prísť až na koniec a jej dĺžka bola iba 3,3125 sekundy.

Pôvodná verzia aplikácie CursorTask obsahovala nasledovné filtre: SpatialFilter, AR Filter, Linear Classifier, LP Filter, Expression Filter a Normalizer. Ako Spatial Filter sme používali malý Laplacian a LP Filter mal ako časovú konštantu nastavenú hodnotu 1s. Ako AR Filter sme používali model 16-teho rádu počítaného na okne dĺžky 0.5s s binmi dĺžky 2Hz pričom ich stredy boli od 1–29Hz. Normalizer pracoval s oknom dĺžky 4s a adaptoval kanál ovládajúci vertikálnu rýchlosť kurzora tak, aby mali hodnoty nulový priemer a rozptyl 1. Pre vysvetlenie filtrov a ich parametrov pozri časť 2.3.2.

Pri kurzorovej aplikácii s LDA klasifikátorom sme používali nasledové filtre: SpatialFilter, AR Filter, LDA Classifier, LP Filter a Normalizer. Prvé dva mali zhodné nastavenia ako v prvom prípade. LP Filter bol zmenený na 0.5s a Normalizer bol prerobený tak, aby produkoval binárny výstup, teda iba čísla -1 alebo 1. Žiadnu normalizáciu nerobil. Filter LDA Classifier bol našou implementáciou LDA klasifikátora popísaného v časti 3.1. Ako vstupné parametre dostával priemery hodnôt pre obe triedy, inverznú kovariančnú maticu, príznaky zvolené pre daný subjekt a typ, akým sa mal adaptovať.

Našim cieľom bolo porovnávať kvalitu zvolených piatich klasifikátorov. Na to sme si museli určiť spôsob ako budeme merať úspešnosť klasifikátora. Aplikácia CursorTask, ktorá bola používaná vo všetkých prípadoch, poskytuje štatistiku o počte cieľov zasiahnutých subjektom. Takáto informácia je dôležitá pre užívateľa, lebo jeho úlohou je trafiť čo najviac cieľov. Budeme ju označovať ako $TaskResult$. Vrchný z cieľov však mohol trafiť úplne hore alebo tesne nad jeho spodným okrajom. Takáto štatistika to vyhlási za trafenie cieľa, ale nehovorí o tom, kde presne ho subjekt trafil. Preto sme doimplementovali druhý typ vyhodnocovania, ktorý v každom kroku klasifikácie zaznamenal, či prebehla správne alebo nie. Z týchto údajov sa potom vypočítala úspešnosť klasifikácie pre každý z cieľov zvlášť. Budeme ich označovať $TargetDown$ a $TargetUp$ podľa umiestnenia cieľa. Po spriemerovaní daných hodnôt sme získali presnejšiu informáciu o celkovej úspešnosti použitého klasifikátora označenú ako $AverageSuccess$.

4.4.3 Predpoklady o výsledkoch

Našou úlohou bolo porovnať úspešnosť rôznych klasifikátorov (popísaných v časti 4.4.2) pri online experimentoch. Z literatúry a vlastných poznatkov sme vedeli, že klasifikátor Standard dokáže podávať slušné výsledky (*TaskResult*) v rozmedzí 70–90%. Očakávali sme, že klasifikátor Supervised by mohol podávať približne rovnaké výsledky, keďže obidva sa učili metódou *učenie s učiteľom*. Supervised však robil klasifikáciu na priestore príznakov, pričom Standard klasifikoval iba jednu hodnotu, ktorá vznikala lineárnou kombináciou príznakov. Preto sme očakávali, že bude Supervised o niečo výkonnejší ako Standard. Keďže sa Unsupervised I a Unsupervised II učili bez znalosti informácie o triede, tak sme očakávali vo všeobecnosti horšie výsledky ako pri metódach typu *učenie s učiteľom*. Chceli sme však vedieť, ako veľmi rozdielne budú výsledky. Keďže sa pri Unsupervised II okrem priemeru adaptuje aj kovariančná matica, tak sme predpokladali, že bude viesť k vyššej úspešnosti klasifikácie ako Unsupervised I. V prípade klasifikátora WithoutAdaptation sme predpokladali, že subjekt bude dosahovať úspešnosť podobnú rozlíšiteľnosti trénovacích dát pri tvorbe LDA (pozri časť 4.3).

4.4.4 Výsledky

Úspešnosť klasifikácie

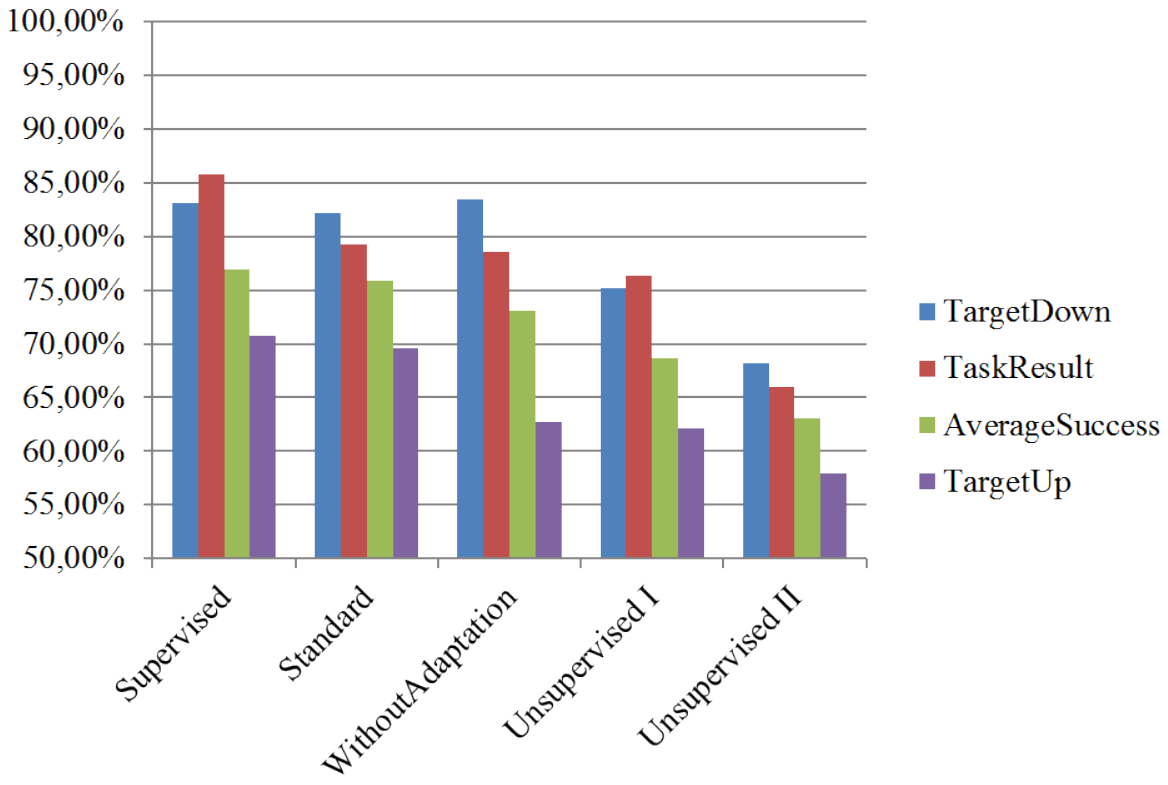
Súhrn všetkých výsledkov online experimenty so spätnou väzbou možno vidieť v tabuľke 4.1 (zaokrúhlené, vyjadrené v %). Zobrazuje výsledky pre všetky typy testovaných klasifikátorov a pre každý z nich hodnotu štyroch sledovaných hodnôt popísaných na konci časti 4.4.2 - výsledok úlohy CursorTask (TaskResult), výsledky klasifikácie pre oba ciele (TargetDown a TargetUp) a ich priemer (AverageSuccess). Hodnoty boli spriemerované pre všetky subjekty, ich všetky sedenia a sady pokusov a sú uvedené v percentách. Riadky tabuľky sú usporiadané podľa výsledkov v stĺpci AverageSuccess zostupne.

Pre vyššiu prehľadnosť o výsledkoch prikladáme obr. 4.7. hodnôt cez všetky klasifikátory, ktoré sú opäť zoradené zostupne podľa výsledkov v stĺpci AverageSuccess. Z výsledkov vyplývajú nasledovné zistenia:

- Poradie klasifikátorov bolo rovnaké vo všetkých prípadoch sledovaných hodnôt okrem TargetDown, kde klasifikátor WithoutAdaptation dosiahol najlepší výsledok.

	TaskResult	TargetDown	TargetUp	AverageSuccess
Supervised	86	83	71	77
Standard	79	82	70	76
WithoutAdaptation	79	84	63	73
Unsupervised I	76	75	62	69
Unsupervised II	66	68	58	63

Tabuľka 4.1: Súhrn výsledkov [v %] online experimentov so spätnou väzbou pre všetky testované klasifikátory.



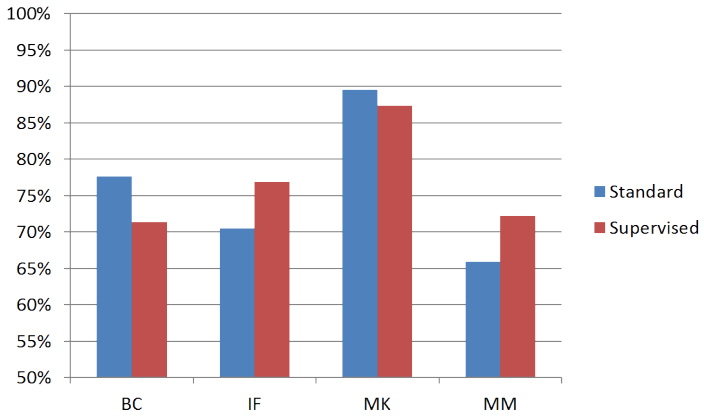
Obr. 4.7: Vizualizácia celkových výsledkov online experimenty so spätnou väzbou.

- Standard podal výsledok, ktorý sme od neho očakávali (70–90%).
- Supervised dosiahol lepšie výsledky ako Standard vo všetkých sledovaných hodnotách. V priemere bol lepší skoro o 4%.
- Napriek očakávaniu dosiahol Unsupervised I lepší výsledok ako Unsupervised II a to priemerne o 8%.

- Lepší z klasifikátorov učených metódou *učenie bez učiteľa* (Unsupervised I) bol v priemere o 5% horší ako klasifikátor Standard.

- Pri klasifikátore WithoutAdaptation dosiahli subjekty lepší výsledok ako bola úspešnosť LDA klasifikátora na trénovacích dátach. Pri sledovanej hodnote *TaskResult* mali všetci vyššiu presnosť a to v priemere o 7%. Pri sledovanej hodnote *AverageSuccess* dosiahla iba polovica subjektov vyšší výkon, avšak po spriemerovaní vyšiel nárast výkonu o 1%.

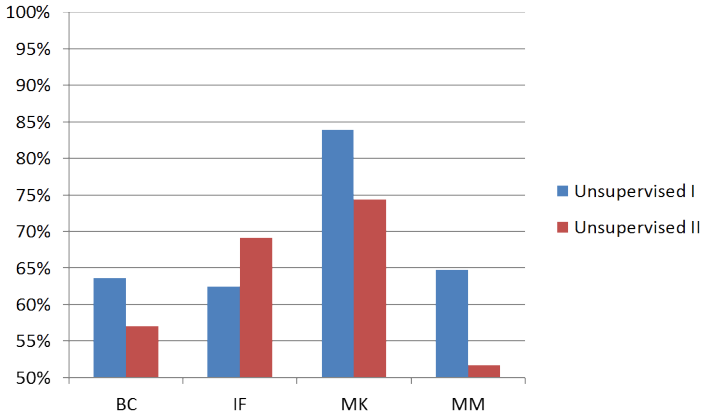
- Pre všetky subjekty vyšla vyššia úspešnosť klasifikácie pre dolný cieľ (TargetDown), ktorý vždy znamenal pohyb rukou. Pre každého bolo ťažšie dostať sa do stavu relaxu (TargetUp). Pre podrobnejšie informácie pozri Prílohu B.



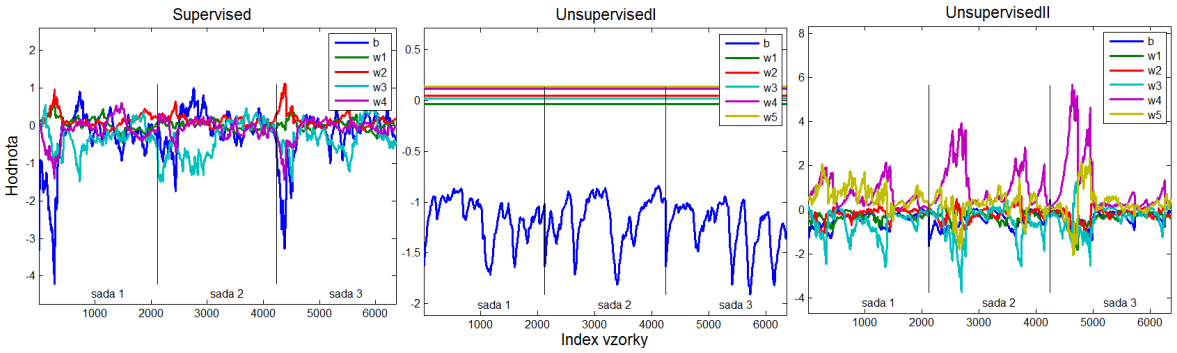
Obr. 4.8: Porovnanie klasifikátorov Standard a Supervised pre všetky subjekty a sledovanú hodnotu AverageSuccess.

Na obr. 4.8 je znázornené, že klasifikátor Supervised dosahoval lepšie výsledky ako Standard iba pre polovicu subjektov, pričom nastalo zlepšenie v priemere o 6,3%. Pre ostatné subjekty došlo k zhoršeniu, pri subjekte MK o 2,2% a pri subjekte BC dokonca o 6,3%. Ako najrelevantnejšiu sledovanú hodnotu sme zvolili priemernú úspešnosť klasifikácie pre oba ciele AverageSuccess.

Pri porovnaní úspešnosti klasifikátorov Unsupervised I a Unsupervised II pre všetky subjekty a sledovanú hodnotu AverageSuccess (obr. 4.9) sme zistili, že Unsupervised II podával horšie výsledky pre 3 subjekty zo 4 a to v priemere až o 9,7%. Iba pre subjekt IF bol Unsupervised II lepší a to v priemere o 6,7%.



Obr. 4.9: Porovnanie klasifikátorov Unsupervised I a Unsupervised II pre všetky subjekty a sledovanú hodnotu AverageSuccess.



Obr. 4.10: Vizualizácia vývoja základných prametrov LDA klasifikátora ($\boldsymbol{w}$ a b) počas všetkých troch typoch jeho adaptácie. Zľava: Supervised, Unsupervised I a Unsupervised II.

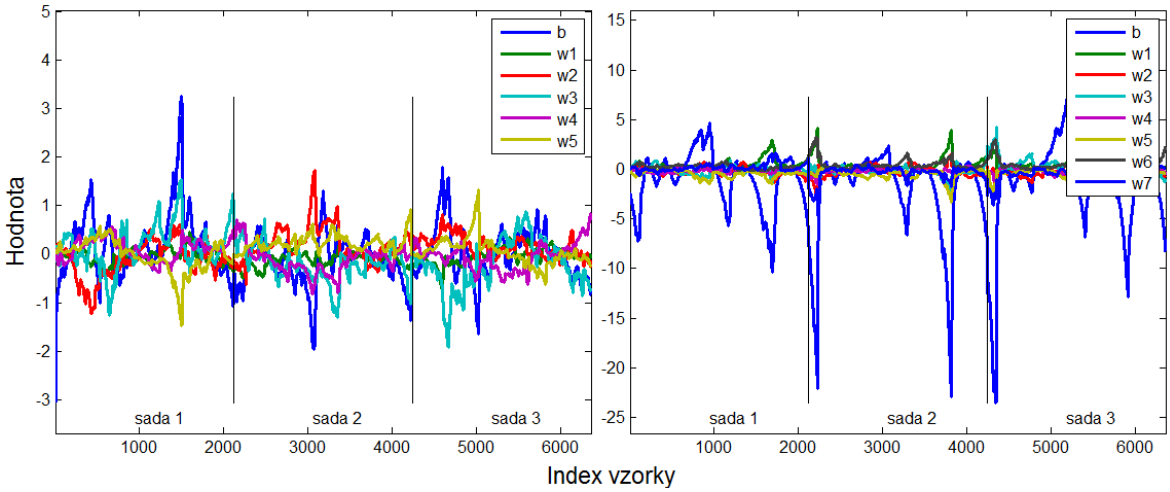
Vizualizácia parametrov klasifikátora

Časťou analýzy výsledkov bola vizualizácia vývoja základných parametrov LDA klasifikátora a to normálového vektora $\boldsymbol{w}$ a hodnoty b. Na obr. 4.10 je možné vidieť vývoj týchto parametrov pre všetky tri typy adaptácie LDA (vybrané sady pokusov pre náhodné subjekty). Takto sme si vizualizovali všetky sady pokusov pre všetky subjekty a z toho vyplynuli nasledovné zistenia:

- Rôzne typy adaptácie vykazovali rôznu dynamiku zmien parametrov. Túto dynamiku sme sledovali pomocou reziduálnej chyby (uvedieme len príklady pre vybraný subjekt a jednu krivku v grafoch - hodnotu b). Najväčšie zmeny prebiehali počas adaptácie klasifikátora Supervised (napr. 0,37), potom v klasifikátore Unsupervised I (napr. 0,244) a najmenšie zmeny vykazoval klasifikátor Unsupervised

II (napr. 0,009).

- Dynamika zmien sa líšila medzi subjektmi. Ukážkou môže byť obr. 4.11, na ktorom vidno, že pri tom istom type klasifikátora môže byť rozdiel v reziduálnej chybe signálu (hodnota b) medzi subjektmi až 17. Chyba vývoja hodnoty b pre subjekt MM bola 0,369 a pre subjekt MK až 17,72.

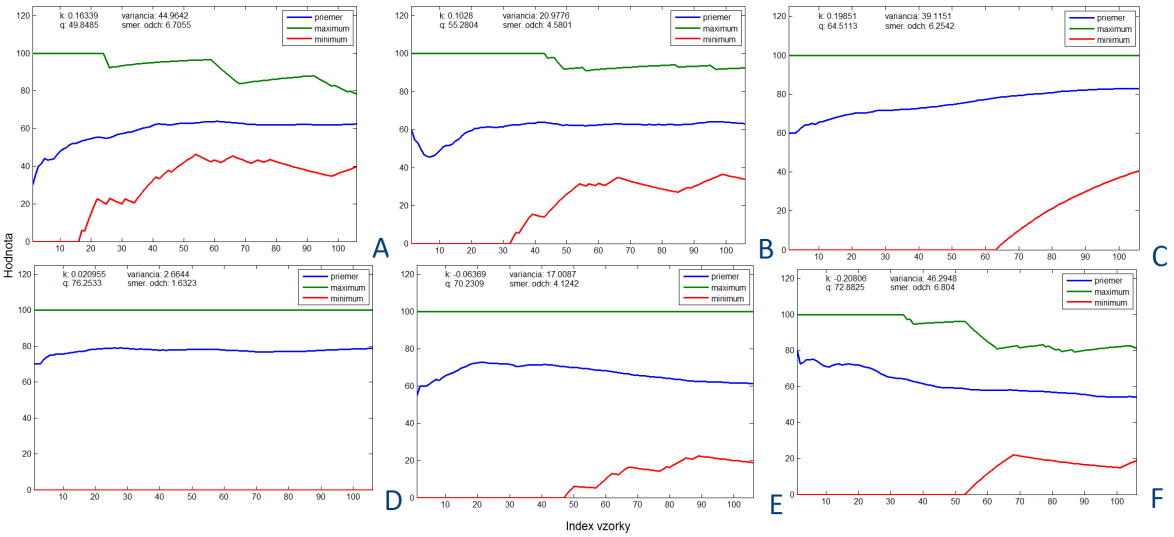


Obr. 4.11: Porovnanie dynamiky zmien parametrov LDA klasifikátora Supervised pre dva rôzne subjekty. Vľavo subjekt MM, vpravo subjekt MK.

Analýza vývoja úspešnosti klasifikácie

Poslednou časťou spracovania výsledkov bola analýza vývoja úspešnosti klasifikácie počas pokusov (jedna cesta guľôčky z ľavého okraja po pravý). Pre každý zo sady pokusov sme vytvorili graf s tromi krivkami: s priemernou, maximálnou a minimálnou krivkou vývoja úspešnosti klasifikácie. Porovnaním takýchto grafov sa však nepodarilo nájsť žiadne opakujúce sa vzory buď pre jednotlivé subjekty alebo pre jednotlivé klasifikátory. Vo väčšine prípadov (75%) úspešnosť klasifikácie počas pokusov stúpala. Klesanie úspešnosti klasifikácie sme mohli najčastejšie sledovať pri Unsupervised II. Tento trend sme zisťovali pomocou lineárnej regresie (Pobočíková and Sedliačková, 2005), ktorú sme aplikovali na priemernú krivku vývoja úspešnosti. Často sa však opakovali niektoré základné priebehy kriviek, ktoré sú zobrazené na obr. 4.12:

- A - subjektu sa na začiatku vôbec nedarilo, ale rýchlo sa mu podarilo zlepšiť a od polovice pokusu sa jeho úspešnosť nemenila.



Obr. 4.12: Najčastejšie sa vyskytujúce priebehy vývoja úspešnosti klasifikácie počas pokusu (rôzne klasifikátory).

- B - najprv išla guľôčka zlým smerom, subjekt sa však začal snažiť a nakoniec sa mu podarilo dosiahnuť správny smer jej pohybu.
- C - úspešnosť klasifikácie rovnomerne stúpala.
- D - úspešnosť bola počas pokusu skoro konštantná.
- E - subjektu sa na začiatku darilo a pravdepodobne, keď zistil, že zasiahne cieľ, tak sa prestal sústrediť a preto úspešnosť klasifikácie počas zvyšku pokusu klesala.
- F - úspešnosť subjektu rovnomerne klesala.

4.4.5 Diskusia

Počas pokusov so spätnou väzbou sme hodnotili úspešnosť klasifikátorov dvomi hodnotami: úspešnosť pokusu (TaskResult) a úspešnosť klasifikácie (AverageSuccess). Z výsledkov sme zistili, že hodnota TaskResult bola vždy vyššia. Pravdepodobne to spôsobuje fakt, že pri hodnote TaskResult nás zaujímal iba výsledok na konci pokusu (či subjekt zasiahol cieľ alebo nie). Počas celého pokusu sa subjektu nemuselo dariť, ale ak sa mu na konci podarilo trafiť aspoň okraj cieľa, tak sa to považovalo za zásah. Pri AverageSuccess sme však vyhodnocovali výsledok klasifikácie v každom kroku a preto je presnejšia. To však spôsobilo jej nižšiu výslednú hodnotu.

Výsledky ukázali, že naučený LDA klasifikátor, ktorý sa však počas experimentu neadaptoval, dokáže podať podobné výsledky úspešnosti ako pôvodne poskytnutý klasifikátor aplikáciou BCI2000. Ak sme takto naučený klasifikátor nechali adaptovať metódou *učenie s učiteľom*, tak to väčšine subjektov pomohlo zlepšiť úspešnosť. Jedine subjekt s označením MK dosiahol vyššiu úspešnosť bez adaptácie. Môže to byť spôsobené tým, že všetky jeho merania prebehli v rozpätí dvoch týždňov. Ostatné subjekty boli merané v rozpätí aj troch mesiacov.

Lepšiu úspešnosť klasifikátora Supervised oproti Standard pripisujeme faktu, že pôvodný klasifikátor z BCI2000 klasifikoval v jednorozmernom priestore a LDA na priestore príznakov, ktorý mal väčšinou 5-7 rozmerov.

Z výsledkov ešte vyplynulo, že subjekty dokázali dosiahnuť s LDA klasifikátorom bez adaptácie lepšie výsledky ako bola úspešnosť toho istého LDA klasifikátora natrénovaného na trénovacej množine (rozlíšiteľnosť dát). Dôvodom tohto javu môže byť fakt, že pri tvorbe trénovacej množiny subjekt nedostával spätnú väzbu o tom, ako veľmi sa mu darilo predstavovať si nejaký pohyb alebo relaxovať. Nevedel teda, či sa má viac snažiť alebo nie. Pri pokusoch so spätnou väzbou keď videl, že sa mu nedarí, tak sa začal viac sústrediť, čo pravdepodobne viedlo k vyššej úspešnosti.

Pri meraniach sme zaznamenávali zmeny $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$, vektora $\boldsymbol{w}$ a hodnoty b. Počas analýzy týchto dát a následnej vizualizácii sme pocítili potrebu sledovať aj hodnotu $\boldsymbol{\mu}$ a kovariančnej matice $\boldsymbol{C}$. Ich priebehy by nám napríklad mohli pomôcť vysvetliť, prečo bol Unsupervised I lepší ako Unsupervised II. Okrem toho nám v grafoch chýbala informácia o tom, či subjekt zasiahol cieľ alebo nie. Vďaka tejto informácii by sme mohli hľadať vzťah medzi dynamikou vývoja parametrov LDA a úspešnosťou trafenia cieľa.

Záver

V našej práci sme úspešne vykonali základný BCI experiment - pohyb kurzora v 1D. Na to sme využili voľne dostupnú aplikáciu BCI2000, ktorú sme hlavne kvôli implementácii vlastného klasifikátora museli dopodrobna preskúmať. Toto skúmanie vyžadovalo čítanie dokumentácie, početné skúšobné experimenty a často aj čítanie zdrojových kódov a ich úpravu. Všetky naše poznatky sme zhrnuli do jednej kapitoly.

Na to, aby sme mohli robiť pokusy so spätnou väzbou, tak sme najprv museli zistiť, či daný subjekt dokáže pomocou predstavovaného pohybu spôsobovať desynchronizáciu μ-rytmu. Jeho úlohou bolo vykonávať množinu úloh, pri ktorých sme zaznamenávali EEG signál. Analýzou signálu sme zistili, či bude ovládanie kurzora možné a ktorý predstavovaný pohyb je pre daný subjekt najvhodnejší.

Okrem toho sa nám podarilo naimplementovať iný druh klasifikácie ako bol pôvodne poskytnutý v aplikácii BCI2000. Z mnohých možností (vrátane nelineárnych metód) sme si vybrali štandardnú metódu *lineárna diskriminačná analýza* (LDA). Vyskúšali sme rôzne typy adaptácie LDA klasifikátora a porovnali navzájom ich úspešnosť. Počas celej našej práce sme urobili dokopy viac ako 60 meraní, ktoré sme zavŕšili ovládaním kurzora pomocou predstavovaných pohybov a relaxácie.

Porovnávali sme úspešnosť piatich klasifikátorov: pôvodne poskytnutý klasifikátor v BCI2000 (Standard), LDA klasifikátor bez adaptácie (WithoutAdaptation), LDA klasifikátor adaptovaný metódou *učenie s učiteľom* (Supervised) a dva klasifikátory učené rôznymi variantami metódy *učenie bez učiteľa* (Unsupervised I a Unsupervised II).

Počas našich experimentov dosiahol najvyššiu úspešnosť klasifikátor Supervised. Bol v priemere o 4% lepší ako Standard. Napriek očakávaniam dosiahol klasifikátor Unsupervised I lepší výsledok ako Unsupervised II a to v priemere o 8%. Lepší z klasifikátorov adaptovaných metódou *učenie bez učiteľa* (Unsupervised I) bol v priemere o 5% horší ako klasifikátor Standard. Všetky subjekty dokázali s klasifikátorom WithoutAdaptation dosiahnuť úspešnosť vyššiu ako bola rozlíšiteľnosť trénovacích dát

použitých pri jeho tvorbe.

Analyzovali sme a následne vizualizovali vývoj základných parametrov LDA klasifikátora počas jednotlivých sád pokusov a to normálový vektor deliacej nadroviny $\boldsymbol{w}$ a posun b. Z priebehu ich vývoja sme zistili rôznu dynamiku ich zmien medzi klasifikátormi a taktiež medzi subjektami.

Nakoniec sme vizualizovali vývoj úspešnosti klasifikácie počas pokusov. Z grafov sa nepodarilo získať žiadne pravidelne sa opakujúce vzory ani pre jednotlivé subjekty, ani pre jednotlivé klasifikátory. Často sa však opakovali niektoré základné priebehy, ktoré sme popísali a vysvetlili ich pravdepodobnú príčinu. Väčšina grafov však vykazovala očakávaný nárast úspešnosti klasifikácie. Žiaducim cieľom v BCI je dosiahnuť minimálne udržanie takejto úspešnosti, v lepšom prípade jej nárast.

Zoznam použitej literatúry

Blankertz, B., Dornhege, G., Krauledat, M., Schröder, M., Williamson, J., Murray-Smith, R., and Muller, K. (2006). The Berlin brain–computer interface presents the novel mental typewriter hex-o-spell. In *Proceedings of the 3rd International Brain–Computer Interface Workshop and Training Course*, pages 108–109.

Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., and robert Müller, K. (2008). Optimizing spatial filters for robust EEG single-trial analysis. In *IEEE Signal Proc. Magazine*, pages 581–607.

Burgess, R. C. (2011). Evaluation of brain connectivity: The role of magnetoencephalography. *Epilepsia*, 52:28–31.

Chaudhary, U., Hall, M., Decerce, J., Rey, G., and Godavarty, A. (2011). Frontal activation and connectivity using near-infrared spectroscopy: Verbal fluency language study. *Brain Research Bulletin*, 84(3):197–205.

Choy, R. and Edelman, A. (2005). Parallel MATLAB: Doing it right. *Proceedings of the IEEE*, 93(2):331.

Cimrová, B. (2011). Vzťah emócie a kognície: pamäťové mechanizmy a emočné udalosti. *Dizertačná práca*, Lekárska fakulta, Univerzita Komenského v Bratislave.

Debnár, T. (2010). Rozhranie mozog–počítač. *Diplomová práca*, Fakulta informatiky a informačných technológií, Slovenská technická univerzita v Bratislave.

Devlaminck, D., Wyns, B., Boullart, L., Santens, P., and Otte, G. (2009). Brain–computer interfaces: from theory to practice. *ESANN'2009 proceedings*, pages 415–424. 22-24 April 2009, Bruges (Belgium).

Hoffmann, A. (2012). Using common spatial patterns for EEG feature selection. MEi:CogSci research project WS 2012, Comenius University in Bratislava.

Krusienski, D. J., Grosse-Wentrup, M., Galán, F., Coyle, D., Miller, K. J., Forney, E., and Anderson, C. W. (2011). Critical issues in state-of-the-art brain–computer interface signal processing. *Journal of Neural Engineering*, 8(2).

Marshall, P. J., Bouquet, C. A., Shipley, T. F., and Young, T. (2009). Effects of brief imitative experience on EEG desynchronization during action observation. *Neuropsychologia*, 47(10):2100–6.

McFarland, D. J., Anderson, C. W., Muller, K.-R., Schlogl, A., and Krusienski, D. J. (2006a). *BCI Meeting 2005–workshop on BCI signal processing: feature extraction and translation.*, volume 14, pages 135–138. IEEE.

McFarland, D. J., Krusienski, D. J., and Wolpaw, J. R. (2006b). Brain–computer interface signal processing at the Wadsworth Center: mu and sensorimotor beta rhythms. *Progress in Brain Research*, 159:411–419.

Neuper, C., Scherer, R., Wriessnegger, S., and Pfurtscheller, G. (2009). Motor imagery and action observation: Modulation of sensorimotor brain rhythms during mental control of a brain–computer interface. *Clinical Neurophysiology*, 120(2):239–247.

Pineda, J. (2005). The functional significance of mu rhythms: Translating "seeing" and "hearing" into "doing". *Brain Research Reviews*, 50(1):57–68.

Pobočíková, I. and Sedliačková, Z. (2005). Lineárna regresia pomocou Matlabu. In *4th International Conference APLIMAT 2005*, pages 351–356.

Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., and Wolpaw, J. R. (2004). BCI2000: a general-purpose brain–computer interface (BCI) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043.

Schalk, G. and Mellinger, J. (2010). *A Practical Guide to Brain–Computer Interfacing with BCI2000*. Springer London.

Vernieri, F., Tibuzzi, F., Pasqualetti, P., Rosato, N., Passarelli, F., Rossini, P. M., and Silvestrini, M. (2004). Transcranial doppler and near-infrared spectroscopy can evaluate the hemodynamic effect of carotid artery occlusion. *Stroke*, 35(1):64–70.

Vidaurre, C., Kawanabe, M., von Bünau, P., Blankertz, B., and Müller, K.-R. (2011). Toward unsupervised adaptation of LDA for brain–computer interfaces. *IEEE Trans. Biomed. Engineering*, 58(3):587–597.

Vidaurre, C., Sannelli, C., Müller, K.-R., and Blankertz, B. (2010). Machine-learning based co-adaptive calibration: A perspective to fight BCI illiteracy. In *Hybrid Artificial Intelligence Systems*, pages 413–420.

Šilar J., Kokoška M., Rebrová K., Farkaš I. (2011). Motor resonance based desynchronization of the EEG mu rhythm. *Activitas Nervosa Superior Rediviva*, 53. Abstract.

Zhao, Q. B., Zhang, L. Q., and Cichocki, A. (2009). EEG-based asynchronous BCI control of a car in 3D virtual reality environments. *Chinese Science Bulletin*, 54(1).

Príloha A - CD

K diplomovej práci prikladáme DVD médium s nasledujúcimi údajmi:

- zdrojové kódy naimplementovaných filtrov a modulov pre BCI2000,
- skripty v jazyku Matlab, ktoré boli využívané pri spracovávaní signálu,
- všetky namerané EEG dáta s popisom, čo bolo merané a používanými parametrami,
- používaná aplikácia BCI2000 v SVN verzii 3800,
- všetky vytvorené vizualizácie adaptácie LDA klasifikátora,
- elektronická verzia diplomovej práce.

Príloha B - Podrobné výsledky

V nasledujúcich tabuľkách uvádzame podrobné výsledky pre experiment so spätnou väzbou. V každej tabuľke môžete vidieť všetky porovnávané klasifikátory a pre nich všetky štyri sledované hodnoty: TaskResult, TargetDown, TargetUp a AverageSuccess. Kvôli diskrétnosti sme naše subjekty označili nasledovne: BC, IF, MK a MM.

		BC	IF	MK	MM	average
Standard	TaskResult	82,4%	73,8%	90,7%	70,3%	79,3%
	TargetDown	82,4%	79,9%	98,3%	68,1%	82,2%
	TargetUp	72,9%	61,1%	80,8%	63,7%	69,7%
	AverageSuccess	77,7%	70,5%	89,6%	65,9%	**75,9%**
WithoutAdaptation	TaskResult	75,0%	72,5%	94,7%	72,2%	78,6%
	TargetDown	75,5%	93,3%	97,6%	67,7%	83,5%
	TargetUp	65,3%	41,0%	80,8%	63,7%	62,7%
	AverageSuccess	70,4%	67,1%	89,2%	65,7%	**73,1%**
Supervised	TaskResult	83,9%	82,2%	91,9%	85,0%	85,8%
	TargetDown	76,7%	82,8%	96,8%	76,1%	83,1%
	TargetUp	66,0%	70,9%	77,9%	68,3%	70,8%
	AverageSuccess	71,4%	76,9%	87,3%	72,2%	**77,0%**
Unsupervised I	TaskResult	74,7%	64,4%	89,4%	76,9%	76,4%
	TargetDown	68,5%	67,5%	92,7%	72,2%	75,2%
	TargetUp	58,7%	57,4%	75,1%	57,2%	62,1%
	AverageSuccess	63,6%	62,4%	83,9%	64,7%	**68,7%**
Unsupervised II	TaskResult	60,3%	77,2%	78,1%	48,3%	66,0%
	TargetDown	60,0%	73,9%	83,0%	55,7%	68,2%
	TargetUp	54,0%	64,4%	65,6%	47,7%	57,9%
	AverageSuccess	57,0%	69,2%	74,3%	51,7%	**63,1%**

Obr. 1: Tabuľka podrobných výsledkov experimentu so spätnou väzbou, hodnoty pre subjekt sú spriemerované cez všetky sady meraní

Metóda / Subjekt	BC	31.1.2013							12.2.2013					6.3.2013			
		Run1	Run2	Run3	Run4	Run5	Run6	average	Run1	Run2	Run3	Run4	average	Run1	Run2	Run3	average
Standard	TaskResult	75,0%	95,0%	60,0%	80,0%	90,0%	90,0%	81,7%	95,0%	75,0%	85,0%	100,0%	88,8%	85,0%	65,0%	80,0%	76,7%
	TargetDown	83,7%	90,2%	64,4%	89,8%	89,5%	84,2%	83,6%	89,6%	79,8%	81,2%	95,2%	86,5%	89,4%	62,2%	80,0%	77,2%
	TargetUp	68,0%	83,7%	57,7%	61,2%	74,4%	79,1%	70,7%	88,7%	63,4%	80,8%	84,4%	79,3%	67,3%	62,1%	76,9%	68,7%
	AverageSuccess	75,8%	86,9%	61,1%	75,5%	82,0%	81,7%	**77,2%**	89,2%	71,6%	81,0%	89,8%	**82,9%**	78,3%	62,1%	78,4%	**73,0%**
WithoutAdaptation	TaskResult	75,0%	75,0%					75,0%	100,0%	90,0%			95,0%	55,0%			55,0%
	TargetDown	44,3%	47,8%					46,1%	86,6%	82,8%			84,7%	95,8%			95,8%
	TargetUp	89,7%	90,9%					90,3%	95,2%	76,9%			86,0%	19,6%			19,6%
	AverageSuccess	67,0%	69,4%					**68,2%**	90,9%	79,9%			**85,4%**	57,7%			**57,7%**
Supervised	TaskResult	90,0%	85,0%					87,5%	80,0%	85,0%			82,5%	85,0%	80,0%	80,0%	81,7%
	TargetDown	79,8%	72,8%					76,3%	80,8%	78,2%			79,5%	87,4%	64,8%	70,5%	74,2%
	TargetUp	72,5%	68,9%					70,7%	58,4%	63,2%			60,8%	68,3%	65,1%	66,3%	66,6%
	AverageSuccess	76,2%	70,8%					**73,5%**	69,6%	70,7%			**70,2%**	77,8%	65,0%	68,4%	**70,4%**
Unsupervised I	TaskResult	65,0%	85,0%	65,0%				71,7%	80,0%	95,0%			87,5%	75,0%	50,0%	70,0%	65,0%
	TargetDown	58,1%	68,5%	60,1%				62,2%	71,3%	81,8%			76,6%	71,0%	61,9%	67,1%	66,7%
	TargetUp	40,1%	65,0%	54,7%				53,3%	65,5%	76,5%			71,0%	54,2%	47,3%	54,2%	51,9%
	AverageSuccess	49,1%	66,7%	57,4%				**57,8%**	68,4%	79,2%			**73,8%**	62,6%	54,6%	60,6%	**59,3%**
Unsupervised II	TaskResult	55,0%	35,0%					45,0%	80,0%	85,0%			82,5%	60,0%	45,0%	55,0%	53,3%
	TargetDown	55,9%	44,5%					50,2%	65,0%	76,6%			70,8%	62,7%	60,3%	54,2%	59,1%
	TargetUp	48,3%	43,1%					45,7%	62,9%	71,4%			67,2%	49,5%	48,0%	49,6%	49,1%
	AverageSuccess	52,1%	43,8%					**48,0%**	64,0%	74,0%			**69,0%**	56,1%	54,2%	51,9%	**54,1%**

Obr. 2: Tabuľka podrobných výsledkov experimentu so spätnou väzbou pre subjekt s označením BC

Subjekt / Metóda	IF	27.2.2013				29.1.2013					20.3.2013				
		Run1	Run2	Run3	average	Run1	Run2	Run3	Run4	average	Run1	Run2	Run3	Run4	average
Standard	TaskResult					85,0%	80,0%	80,0%	90,0%	83,8%	50,0%	75,0%	70,0%	60,0%	63,8%
	TargetDown					83,1%	87,3%	90,8%	97,2%	89,6%	53,6%	90,2%	62,5%	74,6%	70,2%
	TargetUp					71,3%	56,8%	68,7%	78,3%	68,8%	40,1%	61,1%	58,1%	54,6%	53,5%
	AverageSuccess					77,2%	72,0%	79,8%	87,7%	**79,2%**	46,8%	75,7%	60,3%	64,6%	**61,9%**
WithoutAdaptation	TaskResult					90,0%				90,0%	55,0%				55,0%
	TargetDown					88,8%				88,8%	97,7%				97,7%
	TargetUp					74,9%				74,9%	7,1%				7,1%
	AverageSuccess					81,8%				**81,8%**	52,4%				**52,4%**
Supervised	TaskResult	90,0%	80,0%	80,0%	83,3%	80,0%	90,0%			85,0%	85,0%	85,0%	65,0%		78,3%
	TargetDown	84,2%	87,3%	54,3%	75,3%	90,8%	83,5%			87,1%	86,5%	94,2%	77,4%		86,0%
	TargetUp	69,7%	66,4%	74,6%	70,3%	70,6%	89,0%			79,8%	66,9%	69,5%	52,1%		62,8%
	AverageSuccess	76,9%	76,8%	64,5%	**72,8%**	80,7%	86,2%			**83,4%**	76,7%	81,9%	64,7%		**74,4%**
Unsupervised I	TaskResult	45,0%	80,0%	50,0%	58,3%	50,0%	75,0%	80,0%		68,3%	60,0%	75,0%	65,0%		66,7%
	TargetDown	52,4%	71,7%	40,8%	54,9%	60,3%	75,9%	85,7%		74,0%	66,9%	81,2%	72,8%		73,6%
	TargetUp	50,9%	63,8%	46,4%	53,7%	58,1%	65,4%	71,6%		65,0%	51,3%	60,8%	48,0%		53,4%
	AverageSuccess	51,7%	67,7%	43,6%	**54,3%**	59,2%	70,7%	78,6%		**69,5%**	59,1%	71,0%	60,4%		**63,5%**
Unsupervised II	TaskResult	65,0%	85,0%	70,0%	73,3%	90,0%	70,0%			80,0%	70,0%	80,0%	85,0%		78,3%
	TargetDown	74,1%	80,6%	72,1%	75,6%	74,3%	70,8%			72,6%	72,5%	77,0%	71,0%		73,5%
	TargetUp	61,7%	65,8%	64,5%	64,0%	70,5%	64,3%			67,4%	51,2%	69,6%	64,7%		61,9%
	AverageSuccess	67,9%	73,2%	68,3%	**69,8%**	72,4%	67,6%			**70,0%**	61,9%	73,3%	67,9%		**67,7%**

Obr. 3: Tabuľka podrobných výsledkov experimentu so spätnou väzbou pre subjekt s označením IF

Subjekt / Metóda	MK	6.3.2013					12.3.2013				20.3.2013			
		Run1	Run2	Run3	Run4	average	Run1	Run2	Run3	average	Run1	Run2	Run3	average
Standard	TaskResult	100,0%	100,0%	95,0%	100,0%	98,8%	90,0%	95,0%	80,0%	88,3%	90,0%	80,0%		85,0%
	TargetDown	100,0%	99,2%	100,0%	94,4%	98,4%	100,0%	100,0%	99,5%	99,8%	94,3%	99,1%		96,7%
	TargetUp	89,4%	93,8%	90,2%	95,6%	92,2%	73,4%	86,0%	66,2%	75,2%	81,4%	68,7%		75,0%
	AverageSuccess	94,7%	96,5%	95,1%	95,0%	**95,3%**	86,7%	93,0%	82,9%	**87,5%**	87,9%	83,9%		**85,9%**
WithoutAdaptation	TaskResult	95,0%	100,0%			97,5%	95,0%	100,0%	95,0%	96,7%	90,0%	85,0%	95,0%	90,0%
	TargetDown	97,6%	97,9%			97,8%	100,0%	100,0%	98,8%	99,6%	100,0%	91,8%	94,1%	95,3%
	TargetUp	88,5%	95,4%			91,9%	80,9%	95,1%	78,9%	85,0%	68,6%	72,4%	80,5%	73,8%
	AverageSuccess	93,1%	96,7%			**94,9%**	90,5%	97,5%	88,8%	**92,3%**	84,3%	82,1%	87,3%	**84,5%**
Supervised	TaskResult	95,0%	100,0%			97,5%	85,0%	95,0%	85,0%	88,3%	95,0%	90,0%	85,0%	90,0%
	TargetDown	93,1%	99,5%			96,3%	95,7%	99,7%	93,0%	96,1%	99,4%	96,7%	97,8%	98,0%
	TargetUp	82,0%	85,6%			83,8%	69,4%	77,7%	72,2%	73,1%	86,2%	69,5%	74,2%	76,7%
	AverageSuccess	87,5%	92,5%			**90,0%**	82,5%	88,7%	82,6%	**84,6%**	92,8%	83,1%	86,0%	**87,3%**
Unsupervised I	TaskResult	95,0%	95,0%			95,0%	90,0%	80,0%	85,0%	85,0%	95,0%	90,0%	80,0%	88,3%
	TargetDown	88,5%	95,3%			91,9%	94,4%	95,1%	96,7%	95,4%	96,2%	89,7%	86,7%	90,9%
	TargetUp	84,9%	76,6%			80,8%	80,4%	65,4%	75,8%	73,8%	69,8%	72,4%	69,8%	70,7%
	AverageSuccess	86,7%	85,9%			**86,3%**	87,4%	80,2%	86,2%	**84,6%**	83,0%	81,0%	78,3%	**80,8%**
Unsupervised II	TaskResult	70,0%	55,0%			62,5%	85,0%	85,0%	95,0%	88,3%	85,0%	85,0%	80,0%	83,3%
	TargetDown	68,5%	63,6%			66,0%	89,2%	92,8%	87,5%	89,8%	91,0%	96,0%	92,6%	93,2%
	TargetUp	52,8%	48,8%			50,8%	71,8%	76,2%	83,1%	77,0%	65,8%	74,9%	66,6%	69,1%
	AverageSuccess	60,7%	56,2%			**58,4%**	80,5%	84,5%	85,3%	**83,4%**	78,4%	85,5%	79,6%	**81,2%**

Obr. 4: Tabuľka podrobných výsledkov experimentu so spätnou väzbou pre subjekt s označením MK

Metóda \ Subjekt	MM	26.2.2013							13.3.2013				26.3.2013			
		Run1	Run2	Run3	Run4	Run5	Run6	average	Run1	Run2	Run3	average	Run1	Run2	Run3	average
Standard	TaskResult	90,0%	65,0%	65,0%	80,0%	65,0%	60,0%	70,8%	60,0%	70,0%		65,0%	85,0%	65,0%		75,0%
	TargetDown	81,9%	71,0%	64,8%	78,8%	66,1%	57,0%	69,9%	60,9%	68,8%		64,9%	77,4%	61,5%		69,4%
	TargetUp	86,5%	52,3%	64,1%	74,4%	62,2%	47,9%	64,6%	67,0%	65,3%		66,1%	71,6%	49,3%		60,5%
	AverageSuccess	84,2%	61,7%	64,4%	76,6%	64,2%	52,5%	**67,2%**	64,0%	67,0%		**65,5%**	74,5%	55,4%		**65,0%**
WithoutAdaptation	TaskResult	70,0%	70,0%					70,0%	75,0%	65,0%	90,0%	76,7%	75,0%	50,0%	85,0%	70,0%
	TargetDown	81,7%	79,3%					80,5%	59,7%	51,7%	82,4%	64,6%	57,6%	52,2%	64,2%	58,0%
	TargetUp	36,8%	47,8%					42,3%	74,5%	76,6%	90,4%	80,5%	66,7%	55,2%	83,0%	68,3%
	AverageSuccess	59,2%	63,6%					**61,4%**	67,1%	64,2%	86,4%	**72,5%**	62,2%	53,7%	73,6%	**63,1%**
Supervised	TaskResult	80,0%	90,0%					85,0%	85,0%	85,0%	85,0%	85,0%	85,0%	75,0%	90,0%	83,3%
	TargetDown	85,8%	64,8%					75,3%	58,8%	83,6%	73,0%	71,8%	76,1%	80,6%	87,3%	81,3%
	TargetUp	66,1%	72,0%					69,1%	83,6%	57,8%	72,5%	71,3%	74,0%	54,0%	66,0%	64,7%
	AverageSuccess	76,0%	68,4%					**72,2%**	71,2%	70,7%	72,7%	**71,5%**	75,0%	67,3%	76,7%	**73,0%**
Unsupervised I	TaskResult	80,0%	75,0%					77,5%	80,0%	75,0%	65,0%	73,3%	75,0%	85,0%	80,0%	80,0%
	TargetDown	80,4%	70,8%					75,6%	70,7%	70,2%	65,5%	68,8%	69,0%	83,9%	64,0%	72,3%
	TargetUp	66,0%	61,9%					64,0%	59,8%	60,0%	50,4%	56,7%	50,5%	55,8%	46,9%	51,0%
	AverageSuccess	73,2%	66,3%					**69,8%**	65,2%	65,1%	57,9%	**62,8%**	59,7%	69,8%	55,4%	**61,7%**
Unsupervised II	TaskResult	45,0%	55,0%	35,0%				45,0%	50,0%	45,0%	40,0%	45,0%	60,0%	70,0%	35,0%	55,0%
	TargetDown	54,9%	57,4%	53,5%				55,2%	57,9%	53,9%	57,3%	56,4%	56,1%	62,8%	47,4%	55,4%
	TargetUp	56,9%	47,8%	40,1%				48,3%	46,1%	33,9%	47,2%	42,4%	56,8%	53,0%	47,8%	52,5%
	AverageSuccess	55,9%	52,6%	46,8%				**51,8%**	52,0%	43,9%	52,2%	**49,4%**	56,5%	57,9%	47,6%	**54,0%**

Obr. 5: Tabuľka podrobných výsledkov experimentu so spätnou väzbou pre subjekt s označením MM