Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica

Sistema di Controllo dell'Attenzione del Guidatore per mezzi pesanti basato su elettroencefalografia

Laureando

Mauro Bagatella

Relatore

Prof. Nicola Zingirian

Anno Accademico 2015/2016

Indice

Sommario

Il presente lavoro di tesi si prefigge l'obiettivo di progettare e predisporre un sistema di rilevamento ed elaborazione di dati generati da un dispositivo elettroencefalografico. Tale progetto è finalizzato all'identificazione automatica di pattern comportamentali di un conducente alla guida di un mezzo su strada allo scopo di determinare il livello di attenzione prestato dal soggetto dell'esperimento stesso e portare ad una risposta preventiva in termini di riduzione del rischio d'incidente.

Capitolo 1

Introduzione

1.1 Motivazioni della Tesi

Gli incidenti tendono a ripetersi perché, spesso e purtroppo, gli esseri umani non hanno memoria e dimenticano quanto già accaduto. Inoltre, capita frequentemente che gli incidenti mancati (o i "quasi incidenti") vengano ignorati o sottovalutati.

Le tipologie d'incidente che si possono verificare possono essere raggruppate in tre macro categorie. La prima contenente tutti gli incidenti mortali, che, per l'appunto, comportano il decesso dei soggetti coinvolti nell'incidente. La seconda categoria è definita dagli incidente con lesioni, la quale comprende qualsiasi accadimento che abbia avuto conseguenze fisiche o psichiche sull'uomo. Infine la terza categoria contiene l'incidente nella sua forma più lieve, ovvero l'incidente nella quale i soggetti coinvolti non subiscono alcuna lesione.

Un'analisi più approfondita ci permette di identificare i sopraccitati "quasi incidenti" (in inglese Near Miss). Essi rappresentano degli eventi che avrebbero potuto causare un danno ma che, solo per puro caso, non hanno avuto conseguenze. È interessante notare che tale evento è un mancato incidente oggi, ma domani può trasformarsi in un incidente vero e proprio.

Infine è importante definire anche i comportamenti a rischio d'incidente, i quali sono quella serie di comportamenti che pregiudicano la sicurezza alla guida. Questi possono comprendere distrazioni e leggerezze da parte del conducente di un autoveicolo, oltre all'assunzione di alcool o sostanze psicotrope.

Uno tra i paradigmi del moderno approccio alla sicurezza, il quale mette in relazione le varie categorie d'incidente appena descritte, è costituito dalla cosiddetta "Piramide di Heinrich" (studio effettuato nel 1931 da Herbert William Heinrich [1]).

Dallo studio condotto è emerso che, statisticamente, ad ogni incidente mortale (raffigurato in cima alla piramide nella Figura 1.1) corrispondono:

- 30 Eventi incidentali con lesioni
- 300 Incidenti
- 3.000 Quasi incidenti
- 300.000 Comportamenti a rischio d'incidente

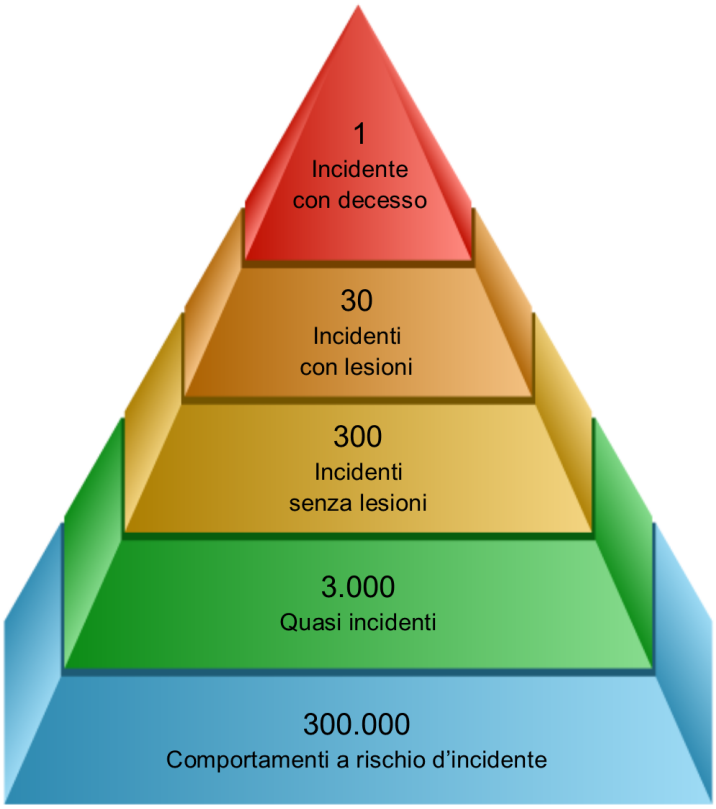

Figura 1.1: Piramide degli incidenti di Heinrich

Come si può vedere alla base della piramide sono stati inseriti i comportamenti a rischio d'incidente. Come si può dedurre la riduzione di dimensione della base di tale piramide porta conseguentemente ad una riduzione di ogni suo livello. In altre parole una diminuzione del numero dei comportamenti a rischio porta ad una riduzione del numero di incidenti e conseguentemente ad un restringimento della quantità di decessi dovuti a tali incidenti.

In quest'ottica di diminuzione del numero degli incidenti partendo dalle cause, vi è quindi la necessità di riconoscere, e dunque prevenire, tutti quei comportamenti considerati a rischio d'incidente. Tuttavia l'identificazione della totalità di tali comportamenti resta tutt'oggi una sfida aperta.

1.2 Obiettivi della Tesi

L'obiettivo della tesi è quello di progettare e predisporre un sistema atto all'elaborazione dei dati generati da un elettroencefalografo finalizzato all'identificazione automatica di alcuni pattern comportamentali di un soggetto alla guida di un automezzo allo scopo di identificare il livello di attenzione e comportamenti a rischio d'incidente.

Più nel dettaglio, gli obiettivi della tesi sono:

1. Valutare la fattibilità dell'integrazione di un brain scanner a basso costo in un sistema di telecontrollo flotte;

2. Proporre un prototipo di sistema in grado di rilevare automaticamente ed in tempo reale le variazioni di tracciati EEG per ciascun canale campionato;

3. Stabilire preliminarmente se le variazioni di tracciato rilevate automaticamente possano essere significativamente correlabili a situazioni di comportamenti a rischio d'incidente da parte di un soggetto alla guida.

Si sottolinea che gli obiettivi proposti non attengono alla rilevazione di patologie né si inscrivono nel perimetro di alcun altro tipo di analisi medica dei tracciati EEG ma riguardano unicamente la rilevazione di variazioni di segnali macroscopici legate agli stimoli che un soggetto impegnato alla guida di un autoveicolo può ricevere.

1.3 Com'è organizzata la Tesi

In questa sezione viene fornita una breve panoramica di com'è strutturata la tesi e cosa in particolare verrà approfondito nei capitoli successivi.

Nel Capitolo 2 verranno definiti i dati necessari all'esperimento che si vuole affrontare ed inoltre verranno descritte brevemente le metodologie con la quale sono stati condotti gli esperimenti in questo lavoro di tesi magistrale.

Nel Capitolo 3 verrà descritto l'hardware del sistema utilizzato in fase di acquisizione dati. Verranno fornite nozioni fondamentali di elettroencefalografia necessarie a comprendere i dati prodotti da un elettroencefalografo (elettroencefalogramma). Inoltre saranno effettuate delle considerazioni sulla scelta del Brain Scanner.

Nel Capitolo 4 verrà descritto come acquisire dati dall'elettroencefalografo utilizzato e le elaborazioni preliminarmente effettuate durante l'acquisizione. Inoltre viene approfondito come ottenere il flusso video da videocamere e come si è potuto sincronizzare tali informazioni con quelle prodotte dal Brain Scanner.

Nel Capitolo 5 verrà descritto quali test sono stati effettuati ed il dettaglio delle condizioni e situazioni nelle quali i dati sono stati ottenuti.

Nel Capitolo 6 si andrà ad effettuare, dapprima, una classificazione manuale degli eventi registrati, successivamente sulla base di questa etichettatura dei dati si compieranno degli studi statistici allo scopo di individuare un metodo di classificazione automatica.

Nel Capitolo 7 verranno approfondite le tecniche di Data Mining utilizzate sia sui dati direttamente registrati dal sistema di acquisizione realizzato, sia su quelli classificati manualmente.

Nel Capitolo 8 verranno fatte alcune considerazioni sul lavoro svolto e identificati margini di miglioramento.

Capitolo 2

Metodologia

In questo capitolo verranno descritte brevemente le metodologie con le quale sono stati condotti gli esperimenti in questo lavoro di tesi magistrale. In particolare verranno definiti i dati necessari allo studio, il metodo con il quale i dati sono stati raccolti ed infine quali analisi sono state poi effettuate utilizzando i dati raccolti.

2.1 Definizione dei Dati

I dati necessari allo sviluppo di questo studio sono rappresentati da:

- Le immagini riguardanti la strada, nonché tutti quelli elementi esterni alla cabina del guidatore;
- Le immagini riguardanti il soggetto alla guida dell'automezzo;
- I segnali prodotti dal dispositivo elettroencefalografico.

Le immagini riguardanti l'esterno della cabina saranno necessarie per catalogare e classificare eventi esterni al guidatore, identificare comportamenti di altri autoveicoli e per effettuare un'analisi ambientale.

Le immagini riguardanti l'interno della cabina saranno fondamentali per catalogare e classificare tutti quegli eventi direttamente collegati al guidatore, identificandone i movimenti e le interazioni concentrate principalmente nelle zone del volto e del capo.

I dati provenienti dall'elettroencefalografo, rappresentanti il cuore dell'esperimento, sono indispensabili per capire quali variazioni dell'elettroencefalogramma intercorrono ad ogni evento identificato dai due dati precedenti.

2.2 Acquisizione dei Dati

Per ottenere i dati sopra definiti è stata necessaria la creazione di un sistema che acquisisca i dati, mantenendone la congruenza e la sincronia allo scopo di permettere una successiva analisi dei dati così raccolti. Tale sistema è composto da due videocamere, opportunamente posizionate all'interno di un automezzo, per permettere l'acquisizione delle immagini dell'autista e della strada, mantenendo la sincronia tra i due flussi video. Inoltre il sistema è stato equipaggiato con un elettroencefalografo, anch'esso in sincronia con le due videocamere del sistema, il quale ha permesso l'acquisizione del segnale elettroencefalografico.

2.3 Analisi dei Dati

I dati raccolti, ed opportunamente elaborati (come si vedrà in dettaglio nei capitoli successivi), sono poi stati classificati manualmente, osservando le registrazioni provenienti dalle due telecamere ed associando ad ogni evento rilevante i relativi dati registrati dall'elettroencefalografo.

Una volta completata la classificazione manuale degli eventi sono stati effettuati degli studi statistici (i quali verranno illustrati nel Capitolo 6) allo scopo di rilevare quali canali del Brain Scanner risultano essere i più rilevati per quanto riguarda la misurazione del grado di attenzione, per poi costruire un classificatore automatico di tali dati in base agli studi condotti.

2.4 Analisi automatica dei Dati

Una volta effettuata una classificazione manuale dei dati si è cercato un modo per automatizzare e rendere tale procedimento funzionante in tempo reale. A tal proposito si è costruito un classificatore automatico in grado di etichettare i differenti pattern con classi predefinite.

Per perseguire tale obiettivo sono state seguite più strategie. In particolare, è stato costruito un algoritmo decisionale basato su regole per realizzare tale classificatore. La sua descrizione la si può trovare nel Capitolo 6. Inoltre si sono applicate tecniche di Data Mining al dataset costruito. Sono stati testati diversi noti algoritmi di costruzione di modelli di classificazione. I dettagli e le performance del modello così costruito sono descritti nel Capitolo 7.

Capitolo 3

Strumentazione

3.1 Elettroencefalografo

L'elettroencefalografo è uno strumento che, mediante elettrodi applicati al cuoio capelluto, viene utilizzato per eseguire e registrare l'elettroencefalogramma, vale a dire per rilevare i potenziali elettrici del cervello e trasformarne l'andamento temporale in una forma grafica (EEG) [2].

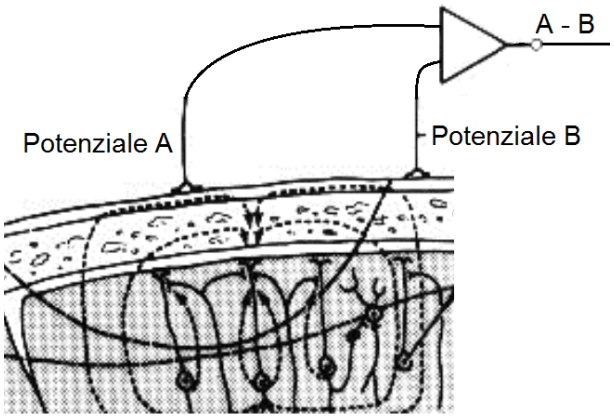

Figura 3.1: Potenziale elettrico

L'apparecchiatura è in grado di registrare i potenziali elettrici dei neuroni della corteccia cerebrale mediante l'utilizzo di elettrodi che vengono applicati in precise aree del capo. Tali elettrodi sono connessi a un sensore di riferimento che funge da massa (chiamato, appunto, elettrodo di massa). È possibile, quindi, calcolare la differenza di potenziale mediante il confronto del potenziale registrato dagli elettrodi applicati e l'elettrodo di riferimento (vedi Figura 3.1). Il segnale raccolto dagli elettrodi è d'intensità insufficiente a compiere rilevazioni significative, quindi viene inviato, di norma, ad un amplificatore differenziale, il cui scopo è

quello di amplificare la tensione tra l'elettrodo attivo e quello di riferimento [3].

Nello sviluppo di questo progetto sono stati tenuti in considerazione due elettroencefalografi, ovverosia Emotiv EPOC^{TM}ed Emotiv INSIGHT^{TM}. Sebbene il secondo sia l'ultimo modello prodotto dalla casa e abbia diversi vantaggi rispetto al primo, che andremmo a descrivere di seguito, gli esperimenti sono stati condotti utilizzando il primo.

La principale motivazione che ha condotto alla scelta di questi particolari dispositivi, a differenza di dispositivi medici più professionali, è il costo contenuto. Ciò consente a chiunque di acquistarlo per uso personale oppure di acquistarne in maggiori quantità per riuscire a monitorare più soggetti contemporaneamente.

Di seguito verrà descritto più nel dettaglio il funzionamento dei due dispositivi e le motivazioni per la quale si è scelto di usare uno rispetto all'altro.

3.1.1 EPOC^{TM}e Windows

Emotiv EPOC^{TM}(Figura 3.2) è uno elettroencefalografo dotato di 14 elettrodi e 2 canali di riferimento e riesce a catturare 128 campioni al secondo per ogni canale. Tali campioni vengono trasmessi come numeri in virgola mobile approssimativamente su un intorno di 4200 microvolt con margine d'errore del convertitore analogico-digitale di 0.51 microvolt [4].

Sebbene il SDK (Software Development Kit) dell'EPOC permetta di sviluppare software che interagiscano con il brain scanner in diversi linguaggi (Java, Python, .Net e C++), il suo supporto è limitato alla piattaforma Windows, dato che le librerie fornite sono sviluppate solo per tale Sistema Operativo.

L'EPOC viene connesso al calcolatore attraverso un'interfaccia USB dotata di antenna wireless. Questo aspetto è di particolare importanza dato che ciò garantisce la portabilità del dispositivo permettendo la possibilità di condurre studi al di fuori di un laboratorio, riducendo inoltre l'ingombro di cavi ed estendendo il raggio d'azione per chi indossa tale dispositivo.

3.1.2 INSIGHT^{TM}e Android

Emotiv INSIGHT^{TM}(Figura 3.3) è uno elettroencefalografo fornito di 5 elettrodi e 2 canali di riferimento il quale ottiene i dati cerebrali con una frequenza di campionamento pari a 128 campioni al secondo, i quali vengono rappresentati in virgola mobile come per il precedente dispositivo [5].

Figura 3.2: Emotiv EPOC^{TM}

Il supporto di Emotiv per quanto riguarda questo dispositivo è molto più ampio rispetto a quello riservato per EPOC, dato che i software che comunicano con INSIGHT possono essere sviluppati per diverse piattaforme quali Windows, Linux, MAC OSX, Android e iOS. In particolar modo nel caso di scelta di tale dispositivo si sarebbe optato per lo sviluppo nella piattaforma Android, dal momento che, così facendo, sarebbe stata possibile l'integrazione con altri progetti in corso ed i relativi software avrebbero potuto essere installati anche su dispositivi mobili dal costo contenuto.

INSIGHT può essere connesso sia mediante la stessa interfaccia USB dotata di antenna wireless di EPOC, sia mediante Bluetooth, utile nel caso si fosse deciso di sviluppare una applicazione per la piattaforma Android. Un'altra differenza è data dalla tecnologia a base di polimeri idrofili sfruttata dai sensori di INSIGHT che, a differenza dei sensori basati su soluzione salina di EPOC, non necessitano di alcuna manutenzione.

Per accedere ai dati EEG di tale dispositivo è richiesta una sottoscrizione al servizio Emotiv, la quale non viene richiesta per quanto riguarda EPOC.

Figura 3.3: Emotiv INSIGHT™

3.1.3 EPOC vs INSIGHT

Di seguito viene riportata una tabella riassuntiva delle caratteristiche di EPOC e di INSIGHT.

	EPOC™	**INSIGHT™**
Numero Canali	14	5
Trasmission Rate	128 campioni/s	128 campioni/s
Connessione	USB	USB/Bluetooth
Durata batteria	14 ore	4 ore

Come si può osservare dalla tabella EPOC offre una migliore durata della batteria, utile per poi andare ad effettuare esperimenti duraturi ricaricandola solo una volta al giorno. Inoltre il maggior numero di canali di EPOC ed il fatto che l'accesso ai dati EEG sia limitato e a pagamento nel caso di INSIGHT ci ha portati a scegliere il primo dei due dispositivi nei nostri esperimenti.

3.1.4 EPOC: Segnale EEG

In questo paragrafo viene mostrato un esempio concreto del tracciato di un elettroencefalogramma (EEG) prodotto da EPOC. Il segnale EEG, come già accennato, consiste in una serie di potenziali elettrici variabili nel tempo (espressi in microvolt) derivanti da diversi canali. Ogni canale rappresenta fisicamente un elettrodo piazzato sul cuoio capelluto. La Figura 3.4 mostra un segnale EEG di 10 secondi ottenuto direttamente dall'Emotiv EPOC (1280 campioni per ogni

canale). Inoltre nella Figura 3.5 si può vedere come sono mappati i vari canali una volta posizionato il Brain Scanner.

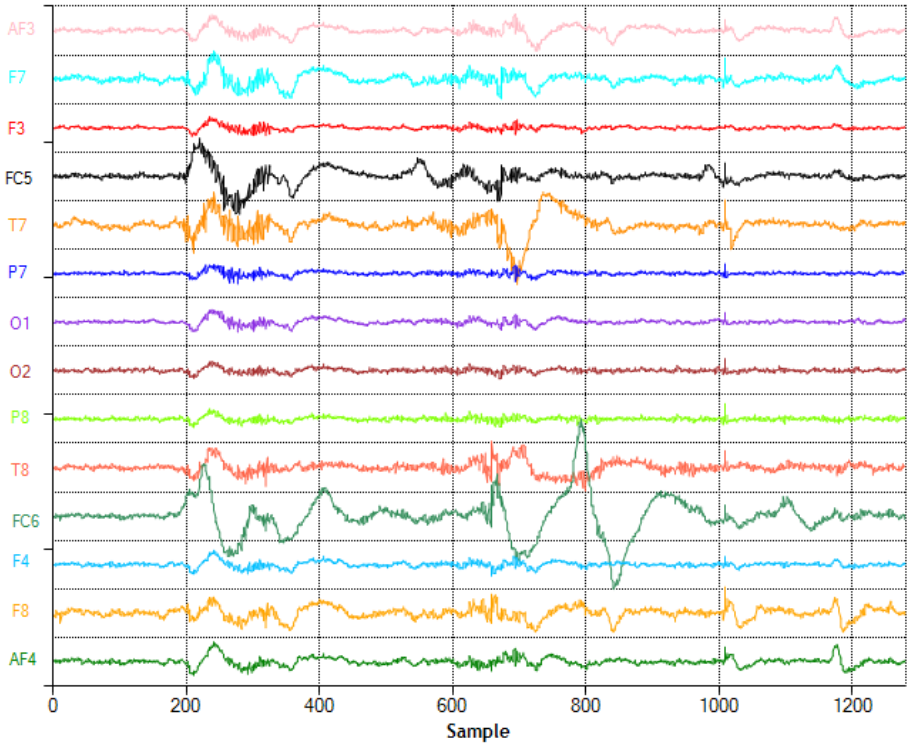

Figura 3.4: Segnale EEG generato da EPOC™

3.1.5 Filtro sviluppato

Il lavoro sviluppato in questa tesi poggia le sue fondamenta in un precedente lavoro di tesi triennale, nella quale venne sviluppato un filtro applicato sul segnale dell'EPOC in grado di rilevare automaticamente ed in tempo reale le variazioni di tracciati EEG per ciascun canale campionato.

Già da primi semplici test effettuati si è visto come i valori prodotti dall'elettroencefalografo varino, anche significativamente, per azioni relativamente vicine alla zona facciale. Per esempio movimenti di braccia e gambe portano a variazioni poco rilevanti del segnale registrato, mentre altre azioni localizzate a livello del viso, come la deglutizione, l'ammiccamento e il digrignamento dei denti portano a variazioni significative della forma d'onda. È stato dunque sviluppato un programma filtro dedicato al rilevamento di tali variazioni del segnale cerebrale, il cui funzionamento sarà approfondito in maniera più dettagliata nei capitoli successivi.

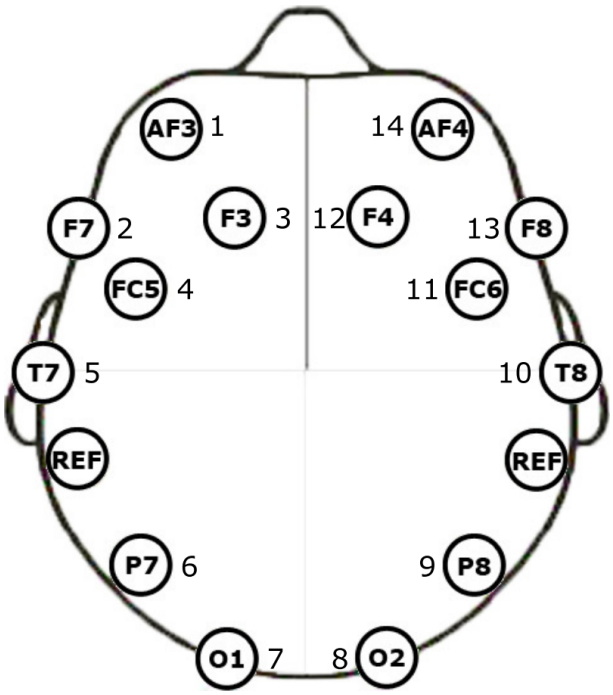

Figura 3.5: Emotiv EPOC^{TM}: Schema di posizionamento dei sensori

3.2 Mini PC

Per poter ottenere i dati dal dispositivo EPOC^{TM}utilizzando le librerie fornite da Emotiv (le quali verranno approfondite più nel dettaglio nel capitolo successivo) è necessario un computer dotato del sistema operativo Windows. Per ridurre l'ingombro nell'abitacolo il più possibile si è optato per un mini pc, un dispositivo moderno e di dimensioni contenute.

Le dimensione del dispositivo in questione sono 12 x 12 x 2,4 cm e di seguito vengono elencate nel dettaglio le sue caratteristiche hardware:

- OS: Windows10 64bit
- CPU: Intel Atom x5-Z8300 (2M Cache, 1.84 GHz)
- RAM: DDR3 2GB
- System Disk: 32GB
- Ethernet: 1000Mbps LAN
- WIFI: IEEE 802.11a/b/g/n, 2.4G+5.8G;

Esso inoltre dispone di 3 porte USB che sono state utilizzate per la connessione del ricevitore wireless del brain scanner e delle due videocamere necessarie alle riprese. È fornito anche di un lettore di schede SD che si è rivelato utile per l'espansione della memoria. Nella Figura 3.6 si può vedere tale dispositivo.

Figura 3.6: Mini PC utilizzato

3.3 Telecamere e Memoria

Per effettuare l'acquisizione del video sono state utilizzate due videocamere USB marchiate AUSDOM®, nella Figura 3.7 si può osservare tale dispositivo. Esso permette di riprendere video ad una risoluzione di 1920 x 1080 pixel e massimo a 30 frame per secondo. Dal momento che i video servivano per la fase di analisi e al solo scopo di capire quali eventi intercorressero in determinati momenti della registrazione, e dal momento che la memoria a disposizione dal computer sopra descritto era alquanto limitata, la risoluzione con la quale sono effettivamente stati catturati i video è stata ridotta a 480 x 360 pixel, ed il frame rate è stato ridotto a 15 fotogrammi al secondo.

Nonostante la compressione dei dati prodotti dall'elettroencefalografo, e nonostante la riduzione di risoluzione e di frame rate del video, la sola memoria di 32GB (12GB effettivamente disponibili) del mini calcolatore risultava essere insufficiente per poter immagazzinare diversi giorni di registrazioni. Per questo motivo si è sfruttata la possibilità di aggiungere una scheda SD all'elaboratore ampliandone così la memoria. La scheda SD installata è stata una Lexar 633x da 64GB.

Dal momento che i video così codificati pesano mediamente 45MB ogni 10 minuti, mentre i dati del brain scanner hanno una dimensione di circa 11 MB ogni ora (una volta compressi), si può arrotondare ad un consumo orario di memoria di

610 MB. Avendo quindi a disposizione una memoria SD da 60 GB (effettivamente disponibili) si ha la possibilità di registrare per circa 100 ore, ovvero 12 giorni lavorativi (calcolando giornate lavorative da 8 ore).

Figura 3.7: Videocamera utilizzata per le riprese

Capitolo 4

Acquisizione e sincronizzazione dei Dati

Questo capitolo si focalizza nella parte di programmazione di questo lavoro di tesi. In particolare verrà descritto nel dettaglio com'è possibile ottenere i dati dall'elettroencefalografo e come ottenere le immagini dalle videocamere, nonché l'elaborazione di tali informazioni e la sincronizzazione dei dispositivi.

4.1 Dati Brain Scanner

In questa sezione verrà descritto più nel dettaglio come accedere ai dati prodotti dall'elettroencefalografo, quali elaborazioni verranno poi fatte su di essi, il metodo con il quale vengono memorizzati e come viene rilevato lo stato del dispositivo.

4.1.1 Accesso ai dati

Una volta connesso il dispositivo di Emotiv al computer mediante l'apposito ricevitore USB wireless, il software sviluppato può iniziare a raccogliere ed immagazzinare dati. Per fare ciò bisogna ricorrere alle librerie fornite da Emotiv stessa. Le API di Emotiv sono accessibili utilizzando 3 header file C (edk.h, EmoStateDLL.h e edkErrorCode.h) e sono implementate in 2 Windows DLL (edk.dll e edk_utils.dll). Il software realizzato, mediante l'utilizzo del linguaggio di programmazione C++, dovrà includere gli header sopra citati ed essere collegato ai file di libreria. Le librerie sono state scritte e compilate utilizzando Microsoft Visual Studio 2005, dunque, per poter scrivere una nuova applicazione che si appoggi su tali librerie occorre utilizzare un prodotto della serie Visual Studio. In particolare, in questo caso è stata utilizzata l'ultima versione (Microsoft Visual Studio 2015). Di seguito vi è una breve descrizione delle principali funzioni utilizzate nel programma sviluppato.

EE_EngineConnect()
Inizializza un'istanza di EmoEngine che permette di leggere dati EEG dall'EPOC.

EE_EmoEngineEventCreate()
Abilita la ricezione degli eventi.

EE_EmoStateCreate()
Crea un buffer nella quale verranno salvati gli eventi.

EE_DataSetBufferSizeInSec()
Crea un buffer, espresso in secondi, nella quale verranno salvati i dati acquisiti.

EE_EngineGetNextEvent()
Legge il prossimo evento nel buffer se presente (un evento potrebbe essere la connessione di un nuovo utente).

EE_DataAcquisitionEnable()
Abilita l'acquisizione di dati per un determinato utente.

EE_DataGetNumberOfSample()
Restituisce il numero di dati presenti nel buffer creato da EE_DataSetBufferSizeInSec().

EE_DataGet()
Permette di ottenere un'intera colonna dei dati presenti nel buffer.

4.1.2 Filtraggio dati

Una volta connessi all'EPOC, creati i buffer, istanziati gli oggetti necessari e abilitata l'acquisizione dei dati mediante le funzioni viste nel paragrafo precedente, si hanno finalmente a disposizione dati provenienti dal Brain Scanner pronti per essere filtrati ed elaborati.

Il filtro realizzato viene applicato ad ogni singolo canale del dispositivo elettroencefalografico e si divide principalmente in quattro operazioni:

1. Normalizzazione dei dati;
2. Filtro Passa Alto;
3. Calcolo energia del segnale;
4. Applicazione della soglia di perturbazione.

In accordo con il primo punto dell'elenco, per prima cosa avviene la normalizzazione dei dati portandoli in un intorno dell'origine, avendo così valori positivi e negativi. Per fare ciò viene semplicemente calcolata la media in un intervallo di larghezza fissa, composto dai valori precedenti all'ultimo dato ricevuto, e sottratta ad esso. Tale operazione viene fatta poiché il valore medio dei dati in uno stato di quiete dipende non solo da soggetto che indossa l'EPOC, ma anche da com'è posizionato e dalla qualità della conduzione, nonché da eventuali sollecitazioni esterne.

Successivamente viene applicato un filtro passa-alto al segnale, il quale permette di eliminare il rumore di fondo e la deriva a lungo termine. La formula utilizzata per il filtro passa alto è

$$y_i = \alpha_1 y_{i-1} + \alpha_2 (x_i - x_{i-1})$$

dove α_1 e α_2 costanti comprese tra 0 e 1, x_i è l'attuale dato da filtrare, x_{i-1} è il dato precedente e y_{i-1} è il valore filtrato precedente [6].

A questo punto, per riuscire ad individuare le perturbazioni del segnale si va a calcolare la sua energia, ovvero la somma dei quadrati dei valori in una determinata finestra. Più il segnale sarà perturbato e più la sua energia sarà elevata. Questa misura viene calcolata per quantificare il livello di attività cerebrale mappato dal canale in questione.

Per finire questa fase di filtraggio si va ad applicare una soglia all'energia appena calcolata. Questa soglia non può essere definita da un valore prefissato, poiché canali differenti produrranno segnali differenti. Inoltre i segnali variano da persona a persona, oltre che da canale a canale. Tale soglia deve essere quindi in grado di adattarsi automaticamente sia a al soggetto sia al canale, evitando però di raggiungere valori troppo elevati che porterebbero a far considerare normali valori anomali, ed evitando di raggiungere valori troppo bassi che porterebbero il filtro ad un'eccessiva sensibilità. Per ovviare a tali problemi viene fornito come input al sistema una soglia massima ed una soglia minima, impostabili sotto forma

di percentuali dell'energia massima registrata. Durante gli esperimenti sono stati utilizzati i seguenti valori: soglia massima 0.8, soglia minima 0.15, α_1 e α_2 a 0.5.

Grazie a tale filtro si riesce dunque a capire quale area dell'encefalo sia in uno stato di attività in ogni istante, e quale invece risulti essere in uno stato di sforzo minore, e dunque considerata in quiete.

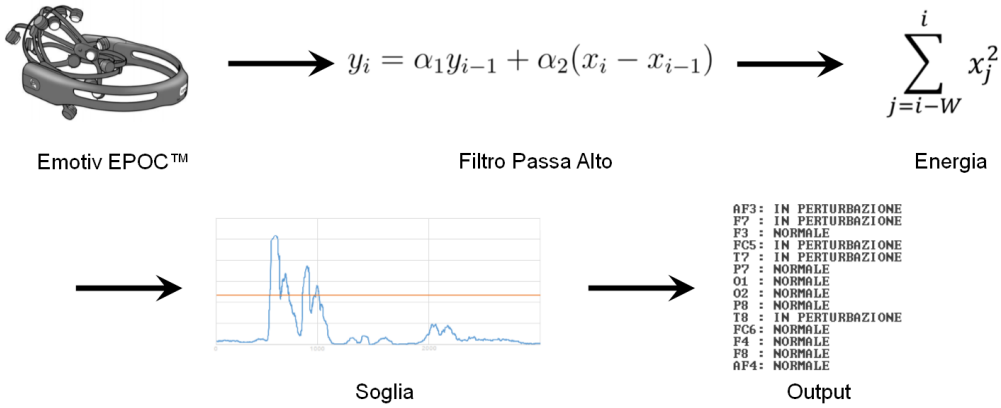

Figura 4.1: Schema del filtro

4.1.3 Identificazione Accensione e Spegnimento

Come visto nei capitoli precedenti il sistema di acquisizione di dati oltre ad essere costituito dall'elettroencefalografo è anche supportato dalla ripresa di videocamere, le quali saranno poi necessarie in fase di analisi dei dati per associare le eventuali attivazioni del filtro utilizzato a specifici eventi accaduti durante l'esperimento.

Per rendere il sistema di acquisizione dati il quanto più automatizzato possibile si è scelto di non far intervenire l'utente per iniziare/fermare le registrazioni. Dunque, l'applicazione realizzata parte non appena il pc viene acceso e resta attiva fino al suo spegnimento, ma a questo punto la registrazione continua da parte delle videocamere risulta superflua, se non dannosa. Superflua, dal momento che si è interessati ad acquisire le immagini dalle telecamere solo nel momento in cui il brain scanner è connesso al sistema, e dannosa poiché nel sistema predisposto si ha a disposizione memoria limitata, dunque una registrazione continua del video porterebbe ad una saturazione prematura della memoria.

Per ovviare a tale problema si è deciso far iniziare la registrazione del video non appena il brain scanner venisse acceso (o connesso al sistema) per poi terminarla non appena esso venisse spento (o disconnesso dal sistema). Purtroppo le

librerie di Emotiv non forniscono tale informazione di accensione/spegnimento. Esse consentono solo di accedere ad un buffer di memoria nella quale vengono salvati i dati dal dispositivo per controllare se ne sono presenti di nuovi, i quali saranno successivamente elaborati e memorizzati dal programma realizzato.

Poiché i dati sono temporizzati da un timestamp si è potuto individuare lo spegnimento del brain scanner calcolando il tempo intercorso dall'ultimo campione ricevuto, ovvero se si supera una soglia temporale, impostata ad 1 secondo, nella quale non vengono ricevuti campioni, il sistema cambia stato riconoscendo il dispositivo come spento e fermando così anche le registrazioni video. Viceversa, se il sistema è nello stato "spento" e rileva che sono presenti dei dati nel buffer dedicato allora passa allo stato di "acceso" iniziando così a raccogliere anche i dati provenienti dalle videocamere.

4.1.4 Compressione Dati

Una volta filtrati i dati proveniente dall'headset si passa alla loro memorizzazione. Per completezza e per ulteriori scopi di analisi si richiede il salvataggio anche di dati EEG prodotti direttamente. Ricordando che il dispositivo produce 128 campioni al secondo per ogni canale, una volta salvati i dati su dico ci si trova di fronte a file dalla dimensione di circa 100MB per ogni ora di registrazione.

I file vengono salvati nel formato CSV (Comma-separated values), e per risparmiare spazio si è scelto di comprimerli non appena l'acquisizione di una sessione è compiuta, riducendo così la dimensione dei file ottenuti di circa 80%-90%, dove la fine di una sessione viene rappresentate dallo spegnimento dell'headset (rilevata come descritto nel paragrafo precedente).

La compressione del file viene effettuata non appena il sistema passa nello stato di "spento" dopo un periodo di attività. Nel caso in cui il computer venisse spento in fase di registrazione, o prima che la fase di compressione giunga al termine, allora si rischierebbe di non comprimere tale file. Dunque, per risolvere tale problema ogniqualvolta inizi una nuova registrazione, viene effettuato un controllo preliminare per verificare se ci sono eventuali file non compressi. In caso affermativo si procede alla compressione, avendo così la garanzia che non ci siano mai file non compressi e sfruttando in questo modo la memoria in maniera più efficiente.

4.2 Acquisizione Video

Nel sistema sono state introdotte delle videocamere (una che punta all'autista dell'automezzo ed una che punta alla strada) allo scopo di monitorare cosa effettivamente stesse accadendo negli istanti in cui il segnale rilevato dal Brain

Scanner andasse ad attivare il filtro. Utilizzando questa procedura è stato necessario trovare un modo per sincronizzare i dati prodotti dal brain scanner con i video in questione.

4.2.1 Registrazione video ed Elaborazione Frame

Per poter accedere alle videocamere è stata utilizzata la libreria OpenCV (approfondita nell'Appendice A) la quale permette, oltre ad accedere ai frame prodotti dalle videocamere, di modificare ogni singolo fotogramma prodotto e di impacchettare tali frame in un file video, per poi codificarlo nel formato desiderato (nel nostro caso MPEG 4).

Come detto in precedenza a questo punto si è dovuto trovare un modo per sincronizzare il video con i dati prodotti dall'elettroencefalografo allo scopo di permettere una successiva analisi di tali dati. Per ovviare a questo problema, utilizzando la libreria OpenCV sopracitata, è impresso su ogni frame il timestamp del momento nel quale è stato acquisito, inserendo ovviamente il timestamp su ogni dato raccolto dal Brain Scanner. Il timestamp è stato registrato nel formato "AAAA/MM/GG HH:MM:SS:mmm".

In dettaglio, è da notare che in fase di registrazione video ogni qualvolta arriva un nuovo frame dalla videocamera viene richiesto al sistema il timestamp mediante la funzione localtime, e una volta convertito nel formato desiderato grazie a *strftime* si procede ad inserire tale stringa, ottenuta dalle funzioni appena citate, nel suddetto frame grazie alla funzione *putText* fornita da OpenCV.

4.2.2 Frame Rate non costante

In questa fase è stato inizialmente riscontrato un problema per quanto riguarda l'acquisizione del video, ovvero il frame rate delle videocamere non risultava essere costante. Si è notato che in base alle condizioni di luce il sensore, in un determinato secondo, poteva produrre più fotogrammi se ben illuminato, o meno fotogrammi se la ripresa avveniva in un ambiente buio. Ciò era dovuto al fatto che le stesse videocamere preprocessassero le immagini prima di mandarle in output. Dal momento che il frame rate viene impostato in maniera costante nell'algoritmo di compressione video in fase di generazione del file, se in determinati istanti ci sono più frame (o meno frame) rispetto al frame rate iniziale allora il video in quei punti si vedrà rallentato (o accelerato).

Per ovviare a tale problema è stata utilizzata una strategia di divisione del tempo in "slot" in funzione degli fps (frame per secondo), dove ad ogni slot temporale viene associato esattamente un frame. È stata dunque costruita una funzione (la quale viene mostrata nell'appendice C.1) che, dato il tempo di arrivo di un frame

e calcolando la differenza di tempo dall'inizio della registrazione e dall'ultimo frame ricevuto, calcolasse esattamente in quale slot tale frame andasse inserito.

Così facendo potevano verificarsi due casi:

1. Nel caso di slot rimasti vuoti (a causa di una diminuzione del frame rate) si ripete il frame precedente fino al riempimento di tali slot.

2. Nel caso di slot già occupato (a causa di un aumento del frame rate) si procede non tenendo in considerazione i frame in eccesso.

In questo modo si è riusciti a stabilizzare i frame rate di qualsiasi telecamera rendendo anche più agevole la fase successiva di reperimento di un esatto istante.

4.2.3 Supporto a più videocamere

Una volta risolto il problema del frame rate non costante si è passati al supporto di più videocamere, permettendo così di ottenere il video derivante da più sorgenti contemporaneamente. Per fare ciò, dopo aver costruito una funzione che si occupasse di effettuare la registrazione video (con la stabilizzazione del frame rate descritta precedentemente) fornendo in input quale fosse la telecamera selezionata, si è passati alla registrazione multipla eseguendo tale funzione su thread separati, uno per ogni videocamera connessa. Tale funzione può essere esaminata nel dettaglio nel frammento di codice C.2.

Come accentato precedentemente gli fps di ciascuna telecamera non sono costanti nel tempo, possono aumentare o diminuire a seconda delle condizioni di luce dell'inquadratura. Dunque, due o più telecamere connesse possono generare un numero di fps differente (sebbene siano dello stesso modello), rallentandosi a vicenda nel caso tutti i fotogrammi venissero catturati nel medesimo thread. Dunque, per rendere indipendenti le registrazioni per ciascuna telecamera si è optato per l'assegnazione di un thread per ciascuna di esse.

4.2.4 Comunicazione con il Brain Scanner

Dal momento che la gestione dei dati proveniente dal Brain Scanner e la gestione del flusso video proveniente dalle videocamere viene affidata a processi separati è sorta la necessità di mettere in comunicazione tali processi allo scopo di sincronizzare anche l'inizio e la fine della registrazione.

Per ovviare a tale problema di sincronizzazione è stato utilizzato un *Named Pipe*, il cui accesso viene fornito direttamente dalla libreria di Windows stessa. I Named Pipe sono dei file speciali che fungono da punto di accesso al pipe, ovvero il

canale di comunicazione tra processi presente nel Sistema Operativo. Una volta generato il file è possibile accedervi in lettura da altri processi. È da sottolineare che in questo caso i dati scambiati non sono memorizzati temporaneamente nel file system, ma transitano da un processo all'altro tramite un buffer.

Non appena l'elettroencefalografo viene acceso, o viene connesso al sistema, il processo che gestisce il flusso dati proveniente da tale dispositivo scrive nel buffer definito dal Named Pipe tale evento. Il secondo processo che si occupa della gestione del video in input delle telecamere controlla costantemente (in un thread apposito) lo stato del pipe per capire quando iniziare (o fermare) la registrazione video. Ovviamente tale procedimento è analogo nel caso di spegnimento del dispositivo, ovvero viene scritto nel buffer un messaggio che identifica lo spegnimento dell'elettroencefalografo dal primo processo. Alla ricezione del messaggio il secondo processo provvede a fermare la registrazione video e ad entrare nuovamente in uno stato d'attesa per l'accensione.

4.3 VideoPlayer

Allo scopo di ottimizzare la ricerca degli eventi nelle registrazioni video è stata sviluppata anche un piccola applicazione che si occupa di cercare determinati istanti e riprodurre il video in quell'istante. Dal momento che si è scelto di spezzare la registrazione video in file da 10 minuti ciascuno, dopo diverse ore di registrazione ci si trova difronte ad una non indifferente quantità di file. Questo porta ad un maggiore lavoro in fase di analisi dei dati, perché se si volesse saltare ad un determinato istante (ad esempio un orario di una determinata giornata) bisognerebbe dapprima individuare il file contenente l'evento ricercato e poi spostarsi al momento corretto, e questo per ogni videocamera.

Dunque per automatizzare questo processo è stato sviluppata un'applicazione chiamata VideoPlayer, ancora una volta sfruttando le librerie di OpenCV. Non appena avviato il software in questione, digitando in input ora e data desiderate esso va automaticamente ad aprire tutti i video che contengono quel determinato istante facendo partire la riproduzione dal momento esatto richiesto. Ovviamente sono stati implementati anche i comandi per riavvolgere il video, metterlo in pausa e scorrerlo frame per frame.

4.3.1 Dettagli del funzionamento

Una volta avviata l'applicazione viene richiesto come input una stringa rappresentate l'istante desiderato nel formato "AAAA-MM-DD HH-MM-SS" (i separatori sono opzionali e al posto di "-" si può inserire uno spazio o qualsivoglia carattere). Non appena l'input viene inserito il software inizia la ricerca del video che

Figura 4.2: Esecuzione di VideoPlayer

contiene tale istante. La ricerca viene effettuata sulla cartella contenente i video, indicata su di un apposito file di configurazione *xml*. Dal momento che si è scelto di nominare i file, in fase di creazione, con l'istante dell'inizio del video secondo questo stesso formato (oltre con id della camera di provenienza) e i video hanno durata massimo di 10 minuti, risulta facile capire in quale file si trovi ciò che si sta cercando.

Per riuscire ad individuare il video ricercato (dopo aver ottenuto la lista di file presenti nella cartella) bisogna valutare se la differenza tra l'istante temporale cercato e l'inizio del video sia inferiore ai 10 minuti, e ovviamente maggiore di 0. A tal proposito è stata costruita una funzione che sfruttando la funzione *difftime* restituisce la differenza di tempo in secondi, la quale, se rientra nel vincolo appena descritto, sarà utile per capire esattamente da quale secondo del video far partire la riproduzione. Sfruttando ancora una volta la libreria OpenCV e la classe *VideoCapture* è possibile mandare in riproduzione il video dall'istante desiderato.

Ovviamente essendoci più telecamere ci saranno anche più video contenenti lo stesso istante temporale, dunque tale programma si occupa anche della riproduzione multipla di tali video, consentendo dunque di recuperali e visionarli tutti contemporaneamente. Nella Figura 4.2 si può vedere l'esecuzione di tale software.

Capitolo 5

Test su Strada

Una volta che il software per l'acquisizione dei dati è stato ultimato si è passati alla fase operativa di raccolta dati mediante l'utilizzo del Brain Scanner unito alle videocamere per la registrazione video.

5.1 Installazione

Una volta avuto a disposizione l'hardware necessario (mini pc, memoria e videocamere) si è proceduto all'installazione del software nel mini pc. Si è configurato il dispositivo in modo che si avviasse automaticamente non appena esso venisse alimentato da corrente per permettere una accensione automatica all'accensione dell'automezzo, il quale soggetto al trasporto di merci pericolose è regolamentato dall'accordo internazionale ADR (Accord Dangereuses Route), il che significa che l'alimentazione elettrica è data solo una volta che la chiave va ad accendere il quadro del mezzo. Per fare ciò si sono dovute modificare alcune impostazioni del BIOS (Basic Input/Output System) del dispositivo, per permetterne l'accensione automatica senza la necessità della pressione del tasto di alimentazione. Inoltre si è impostata l'applicazione sviluppata in modo che venisse eseguita in automatico non appena il PC fosse avviato, sempre allo scopo di garantire il massimo automatismo possibile per quanto riguarda l'acquisizione dei dati, lasciando così l'autista dell'automezzo libero di concentrarsi sulla guida con il minor numero possibile di interferenze al proprio lavoro. Una volta effettuati diversi test per verificare l'effettivo funzionamento di tutto il sistema si è proceduto con l'installazione sull'automezzo.

5.2 Analisi delle risorse utilizzate

Nei vari test effettuati per la verifica dell'effettivo funzionamento del sistema si è andati anche ad analizzare le risorse utilizzate dall'applicazione per capire quanto esosa fosse la registrazione dei dati nel mini pc descritto precedentemente.

Figura 5.1: Installazione sull'automezzo, Lazzarini Moreno (autista dell'automezzo)

Inoltre tale analisi è stata effettuata per garantire l'assenza di Memory Leak dal momento che l'esecuzione di tale applicazione non viene supervisionata da alcun utente in fase di raccolta dati.

L'acquisizione ed il filtraggio del segnale EEG proveniente dal Brain Scanner costa alla CPU un 15% medio di utilizzo, e la porzione di RAM utilizzata rimane fissa a 2,4MB. L'acquisizione del flusso video proveniente dalle due videocamere risulta essere un'operazione più costosa della precedente per l'elaboratore, comunque mantiene buone performance mantenendo un utilizzo della CPU pari al 30% ed occupando mediamente 35MB di RAM.

Generalmente il sistema di acquisizione video/EEG va ad utilizzare un 45% del-

la potenza di calcolo del dispositivo di elaborazione, gravando sulla memoria primaria con meno di 40MB.

5.3 Test su strada

L'azienda di trasporti che ha concesso l'installazione del sistema sviluppato, permettendo così di raccogliere dati, è Tecnotas, azienda facente parte del gruppo Transadriatica. In particolare il dispositivo è stato installato su di una autocisterna dedita al trasporto di carburanti. Si ringrazia inoltre il signor Lazzarini Moreno, l'autista dell'autocisterna che si è prestato all'esperimento permettendo di monitorarlo e di raccogliere dati dalla sua giornata lavorativa.

La raccolta dati prevedeva di monitorare l'autista e rilevare i dati dall'elettroencefalografo durante la fase di guida dell'automezzo nella sua normale giornata lavorativa. Il monitoring del soggetto è durato dal 15 Settembre fino al 20 Ottobre 2016. La giornata lavorativa nel periodo di test è durata mediamente 4 ore e solo in alcuni casi 8 ore. Il numero di giornate lavorative in cui effettivamente sono stati raccolti dati è pari a 22 giorni. Il totale netto delle registrazioni (escludendo momenti in cui l'autista rimane fermo per rifornimenti, carichi e scarichi) è pari a 72 ore e 51 minuti.

È importante sottolineare che i dati sono stati estratti dal dispositivo settimanalmente, e non solamente alla fine del mese, garantendo così la non saturazione della memoria. Inoltre sono state effettuate osservazioni periodiche sul corretto funzionamento dell'intero sistema allo scopo di garantire la massima veridicità e qualità delle informazioni registrate.

Le tratte svolte dall'autista in fase di monitoraggio sono state localizzate in diversi distributori di carburanti principalmente disposti all'interno della regione Veneto. Le quale hanno consentito di ottenere dati di diverse ambientazioni quali: centri abitati, pianure, campagne, colline, montagne, paesi di montagna.

Le fasce orarie sulle quali sono state effettuate le registrazioni sono state capaci di comprendere l'intera giornata:

- Dalle 3:00 alle 8:00
- Dalle 8:00 alle 12:00
- Dalle 12:00 alle 18:00
- Dalle 16:00 alle 22:00

Inoltre in questo arco di tempo è stato possibile registrare una molteplicità di eventi climatici i quali sono stati in grado di variare le rilevazioni: sole, pioggia, nuvoloso, temporali e nebbia.

Tale varietà di location, fasce orarie e condizioni climatiche hanno permesso la costruzione di un dataset completo, permettendo così di analizzare i dati e valutare l'intero sistema sotto diversi punti di vista.

Capitolo 6

Analisi dei Dati

In questo capitolo viene descritto come i dati ottenuti dall'elettroencefalografo, una volta elaborati dal filtro descritto nei capitoli precedenti, vengano classificati manualmente ed analizzati.

6.1 Classificazione manuale dei dati

Una volta ultimata la raccolta dati si è passati alla fase di classificazione ed analisi dei dati. La classificazione manuale è stata effettuata partendo dai dati filtrati dell'elettroencefalografo (con il filtro descritto nel Capitolo 4). Osservando le varie attivazioni dei canali filtrati, e partendo dapprima dagli istanti nella quale un maggior numero di canali risultasse attivo, si è andati ad analizzare, osservando i flussi video registrati dalle due videocamere, a quali eventi corrispondesse ciascuna attivazione. In questa fase è risultato molto utile l'utilizzo dell'applicazione VideoPlayer sviluppata precedentemente, la quale riproduceva esattamente il video riguardante l'istante desiderato partendo dalla data e dall'ora digitata in input, riducendo drasticamente il tempo di ricerca dei file contenenti tale istante (essendo due i file video per ogni istante).

La classificazione manuale è avvenuta identificando 4 differenti classi:

- CLASSE 0: Attivazioni dovute a movimenti del brain scanner o sollecitazioni del capo (ad esempio quando l'operatore si gratta la testa, o si sfrega la fronte).
- CLASSE 1: Attivazioni dovute ad eventi che richiedono un certo livello di attenzione (essi sono, ad esempio, curve strette, incroci, semafori, strade strette, rotatorie, entrate/uscite da autostrade).
- CLASSE 2: Attivazioni dovute ad eventi esterni non particolarmente rilevanti (curve larghe, sorpassi da parte di altri veicoli, salite leggere, discese leggere).

- CLASSE 3: Attivazioni non associate ad alcun evento rilevante (quali cambi di espressione, o possibili deglutizioni).

L'elettroencefalografo è molto sensibile a sollecitazioni esterne, (quando si toccano direttamente i sensori il segnale viene pesantemente influenzato) soprattutto nell'area del volto, quindi ogni volta che l'autista tocca il volto / il capo il segnale è da considerare "corrotto" e quindi viene classificato con 0. Anche quando ci sono particolari vibrazioni del camion e si vede l'autista che oscilla velocemente viene classificato con 0. Quando mastica (soprattutto la gomma da masticare) il segnale viene fortemente influenzato dal continuo masticare e ancora una volta non è utile per le effettive rilevazioni cerebrali quindi anche in questo caso viene classificato con la classe 0.

Tutte le altre classi sono presenti solo quando l'autista si trova in uno stato di quiete e non sta interagendo direttamente con la zona del volto o del capo come descritto precedentemente.

La classe 1 è la classe considerata di maggior interesse, e rappresenta tutti i momenti di effettiva concentrazione da parte dell'autista (sorpassi, partenze da incroci, curve, rotonde, eccetera...).

La classe 2 è rappresenta gli stessi eventi classificati dalla classe 1, ma in questo caso richiedenti meno concentrazione da parte del soggetto alla guida (curve larghe che non richiedono particolare impegno o attenzione).

La classe 3 infine è riservata per eventi non identificati (ovviamente dopo che si è accertati che non si rientra nella 0 e quindi l'autista è in uno stato di quiete) o per eventi non esterni, come ad esempio cambio di espressione facciale, sbattiti delle palpebre o deglutizioni (questa dovrebbe essere la classe più rara).

6.2 Statistiche sui dati classificati

Una volta completata la classificazione manuale degli eventi utilizzando le 4 classi sopra descritte sono stati effettuati degli studi statistici sul dataset di eventi appena creato.

In totale sono stati classificati 1265 eventi dove la seguente tabella riporta il numero di eventi rilevati per ogni classe:

Classe	Numero di eventi
Classe 0	332
Classe 1	647
Classe 2	218
Classe 3	68

6.2.1 Classe di maggior rilevanza

In questa sezione viene mostrato nel dettaglio come sono distribuite le attivazioni dei canali nella classe di maggiore interesse, ovvero nella classe identificata dall'etichetta 1. In particolare la seguente tabella mostra la percentuale di attivazione del canale negli eventi classificati.

Canale	Percentuale di attivazione
AF3	5%
F7	55%
F3	8%
FC5	8%
T7	16%
P7	21%
O1	11%
O2	54%
P8	66%
T8	80%
FC6	68%
F4	56%
F8	73%
AF4	62%

Nella Figura 6.1 si può osservare graficamente la differenza della frequenza di attivazione di ciascun canale, ed in particolare si può notare come i canali presenti in misura maggiore siano quelli che vanno da O2 fino ad AF4.

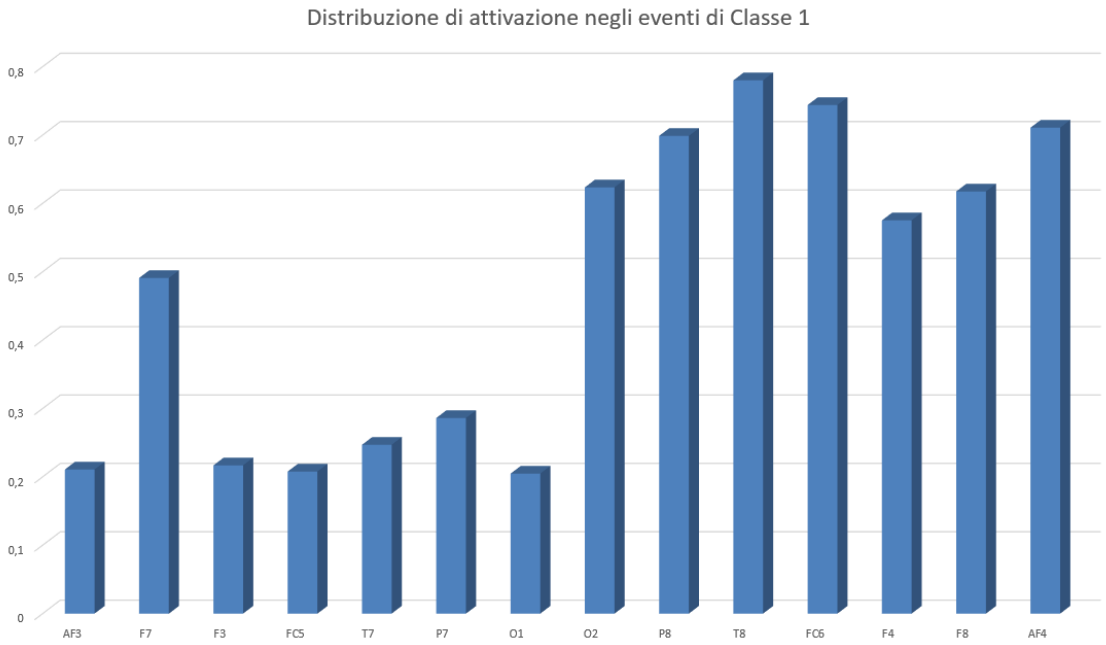

Figura 6.1: Distribuzione di attivazione dei canali

La Figura 6.1 mostra ancora una volta come sono mappati i vari canali. In particolare si può notare che i canali più attivi, ovvero quelli identificati dalle etichette O2, P8, T8, FC6, F4, F8 ed AF4 si trovano tutti nella parte destra dell'encefalo. L' emisfero destro (conosciuto anche come "cervello poeta") è più specializzato nell'elaborazione visiva e nella percezione delle immagini, nella loro organizzazione spaziale e nell'interpretazione emotiva. Più sommariamente, al cervello poeta spetta la percezione globale e complessiva degli stimoli [7].

Risulta dunque interessante notare come gli eventi classificati da tale classe, ritenuta quella di maggior interesse per gli scopi di tale tesi, evidenzi una maggior attività nell'emisfero destro dell'encefalo, ritrovando dunque riscontri con studi medici più approfonditi in tale campo.

Osservando questi risultati si è potuto inizialmente ipotizzare di correlare un certo livello di attività del cervello poeta ad un determinato livello di concentrazione o attenzione del soggetto alla guida dell'automezzo. L'assenza dunque di un determinato grado di attività indicherebbe un scarso livello d'attenzione, e correlando questa informazione alla conoscenza del punto specifico in cui si trova il soggetto si potrebbe determinare il livello d'attenzione richiesto. In questo contesto sarebbe possibile dedurre se l'autista sta dedicando sufficiente attenzione alla manovra che sta compiendo oppure no, in questo caso ci si troverebbe difronte ad un comportamento a rischio d'incidente. Ma prima di affermare ciò è

stato necessario trovare un classificatore automatico in grado di determinare tale classe.

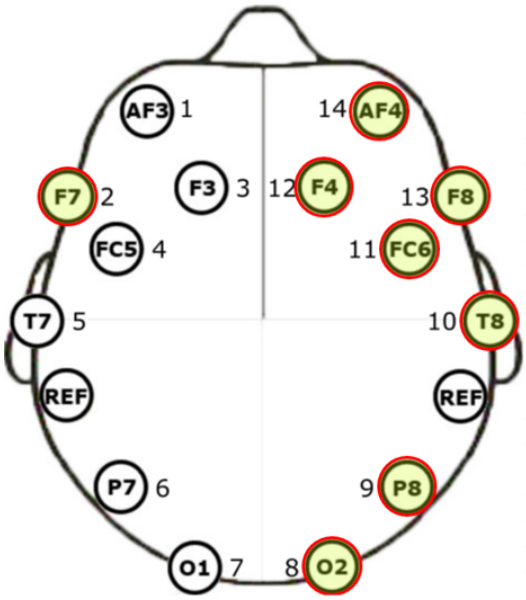

Figura 6.2: Mappatura delle attivazione dei canali

6.2.2 Eventi di classe 0

La maggior parte degli eventi di classe 0 sono identificati da interazioni con il capo da parte dell'autista (dovuti a momenti in cui sistema il posizionamento dei sensori sul capo, o semplicemente a momenti in cui si gratta il capo). Inoltre diverse attivazioni sono dovute a leggeri sobbalzi dell'automezzo, causate da buche o da deformazioni della strada, facendo così sobbalzare di conseguenza anche l'autista.

Le attivazioni dovute a quest'ultimo caso (sobbalzi dell'automezzo) possono essere ridotte o evitante sfruttando anche i dati provenienti dall'accelerometro presente nel Brain Scanner stesso. Dunque se si identificano sobbalzi osservando i dati dell'accelerometro allora si può dedurre che quella sia la motivazione dell'attivazione dei canali, e di conseguenza si può agire con l'inibizione delle attivazioni non appena si rilevano tali sollecitazioni.

Per quanto riguarda le attivazioni dovute al primo caso (interazioni con il capo) si può pensare di rilevarle utilizzando un dispositivo esterno. Ad esempio servendosi di un bracciale dotato di accelerometro e giroscopio si può dedurre la posizione del polso, e dunque della mano. In questo modo si potrebbe capire quando le mani sono distaccate dal volante, ma soprattutto capire quando esse

si trovino nell'area del capo o del volto, associando così le attivazioni dei canali registrate a tale azione.

6.3 Classificazione Automatica

Una volta studiata la distribuzione dei dati si è passati alla realizzazione un classificatore che si occupi di automatizzare il processo di divisione dei dati e riconoscimento degli eventi. In questa fase si è cercato un modo automatico per separare la classe 0 dalle altre classi, per cercare di ridurre il numero di false attivazioni del filtro.

Si è costruito anche in questo caso un applicativo software che si prendesse a carico tutti i dati classificati manualmente e li elaborasse per identificare una strategia di divisione di tali record. Dalla statistica precedentemente effettuata dalla classificazione manuale, operando in un'ottica di divisione degli eventi di classe 0 dagli eventi di classe 1, si ha che la probabilità che un evento sia di classe 1 è pari al 66%, mentre la probabilità che un evento sia di classe 0 è pari al 34%. Su tale base si è calcolato l'errore di classificazione.

È stato dunque realizzato un insieme di classificatori basati su regole, ciascuno di essi finalizzato ad uno scopo. In questo caso gli obiettivi erano due, ovvero minimizzare l'errore in un contesto di falsi positivi (eventi di classe 0 classificati come eventi di classe 1) e massimizzare il numero di eventi di classe 1 riconosciuti correttamente.

6.3.1 Divisione Classe 0 da Classe 1

Il classificatore che minimizza l'errore, ottenendo un errore pari a 0.245442, è stato definito dalla seguente regola:

$$\mathbf{AF3 = 0\ AND\ F3 = 0}$$

La quale sta a significare che se in un record non sono attivi contemporaneamente i canali AF3 e F3 allora esso è rappresenta un evento di classe 1 con probabilità del 75.46%. Mentre la percentuale di record ottenuti applicando tale regola, che sono effettivamente di classe 1, è pari a 90,41% (tale valore viene anche definito Recall). In altre parole con tale classificatore si possono riconoscere oltre il 90% delle situazioni in cui è effettivamente richiesto un certo livello d'attenzione da parte del conducente, ma nel momento in cui il sistema segnala tale evento si ha un 24% di probabilità che esso sia un falso positivo.

Un'altra regola emersa è data dalla semplice:

$$\mathbf{AF3 = 0}$$

Essa riesce ad ottenere il miglior valore di Recall (94.75%) aumentando i falsi positivi al 26.65% ma mantenendo l'errore ad un valore molto simile alla regola precedente (0,248125).

Nel caso si volessero mantenere i falsi positivi ad una percentuale inferiore al 10% la miglior regola che raggiunge questo risultato è la seguente:

$$\mathbf{F3 = 0\ AND\ T7 = 0\ AND\ T8 = 1\ AND\ F8 = 1}$$

Essa permette di avere un valore di false positive rate pari a 9.91% ma riducendo il valore Recall al 49.46% ed aumentando l'errore complessivo a 0.484282.

6.3.2 Divisione Classe 0 da Classe 1 e 2

In questo caso le considerazioni rimangono le medesime considerazioni fatte precedentemente, ma per realizzare tali regole si è cercato di separare la classe 0 dalle classi 1 e 2, fondendo queste ultime in un'unica classe.

Con questo dataset set il classificatore che minimizza l'errore è stato definito dalla semplice regola:

$$\mathbf{AF3 = 0}$$

A differenza del caso precedente la regola che porta l'errore al minimo, pari al valore 0.222943, è data dalla sola condizione "AF3 = 0". In questo caso il valore Recall è pari al 93.41% e il false positive rate è del 21.80%.

Invece la regola "AF3 = 0 AND F3 = 0", che nel caso precedente si trovava al primo posto, in questo contesto porta a risultati, sebbene solo lievemente, peggiori. Essa è caratterizzata da un errore pari a 0.235911, Recall del 88.20% e false positive rate pari a 17.80% (l'unico valore che migliora).

6.3.3 Canali AF3 e F3

Osservando i risultati della classificazione con i relativi errori, si evince che la classe 0 è difficilmente separabile dalle altre di maggior rilievo. I canali che più si prestano a tale separazione sono AF3 e F3. Si deduce, come si vede anche dalla Tabella 6.1 che mostra la perpetuate di attivazione di ciascun canale (avvenuta in tutta la durata dell'esperimento), che l'area del cervello mappata da tali canali non sia particolarmente attiva durante la guida, dunque essi vengo attivati solo

Tabella 6.1: Attivazioni totali dei canali

Canale	Percentuale di attivazione
AF3	0,64%
F7	2,80%
F3	0,77%
FC5	0,76%
T7	1,53%
P7	1,45%
O1	0,46%
O2	2,44%
P8	2,80%
T8	2,75%
FC6	3,09%
F4	2,53%
F8	3,13%
AF4	2,58%

in quei momenti di disturbo nella quale si va a toccare o a muovere tali sensori. Come visto nella sezione precedente i canali attivi maggiormente nella classe 1 si trovano nell'emisfero destro dell'encefalo, ma se i disturbi avvengono in quella stessa zona risulta molto difficile, se non impossibile, identificare se essi siano o no eventi d'interesse senza aver a disposizione maggiori informazioni.

Capitolo 7

Data Mining sui dati

In fase di analisi dei dati sono stati utilizzati vari approcci per aggregare i dati registrati, in particolare in questa sezione si va a descrivere gli approcci di Data Mining utilizzati e i risultati così ottenuti. Per effettuare tale analisi è stato utilizzato il software open source Weka (descritto nell'Appendice B).

7.1 Classificazione automatica

La classificazione automatica dei dati è un processo composto di due passi:

- Costruzione del modello
- Uso del modello

La costruzione del modello viene effettuata a partire da un dataset già classificato, chiamato training set, ovvero da un insieme di dati divisi in record, dove ad ogni record è associata una classe. In questo caso il training set utilizzato è stato quello costruito nella fase di classificazione manuale dei dati.

Il modello, una volta costruito, viene valutato, e per fare ciò esistono diverse misure di valutazione del modello. La misura di valutazione del classificatore più utilizzata è la Matrice di Confusione (vedi Appendice B). Inoltre si ha a disposizione una serie di misure di valutazione più sintetiche della matrice, come ad esempio: precision, recall, true positive rate, false positive rate, true negative rate e false negative rate. La qualità del modello è validata mediate una porzione del training set non utilizzata per la costruzione del modello, definita test set.

Il modello può essere definito mediante alberi di decisione, reti neurali, probabilità, regole di associazione, support vector machines (SVM) o versione ibride di tali modelli di classificazione.

Una volta costruito il modello, ottenendo risultati soddisfacenti dalla fase di valutazione, è possibile passare alla sua effettiva utilizzazione. In questa fase utilizzando il modello appena generato si assegna l'etichetta di classe a nuovi dati non etichettati, operando in tale modo la classificazione automatica di nuovi dati.

7.1.1 Misure di valutazione

In questo paragrafo verranno illustrate nel dettaglio le misure di valutazione del modello di classificazione. Nella Figura 7.1 viene data una rappresentazione grafica di Precision e Recall, nonché dello scenario che si ottiene dopo la costruzione di un modello di classificazione. Di seguito si trovano le formule di tali misure e insieme ad altre di particolare rilievo:

$$Precision = \frac{TP}{TP + FP} \tag{7.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{7.2}$$

$$True\ positive\ rate = \frac{TP}{TP + FN} \tag{7.3}$$

$$False\ positive\ rate = \frac{FP}{FP + TN} \tag{7.4}$$

$$F - Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7.5}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{7.6}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{7.7}$$

Dopo la classificazione (come si vede nella Figura 7.1), a ogni record viene assegnata una delle seguenti etichette: vero positivo, falso positivo, vero negativo e falso negativo. I valori TP, FP, TN e FN che si possono osservare nelle equazioni precedenti si riferiscono alla quantità di record etichettati, rispettivamente, come veri positivi, come falsi positivi, come veri negativi e come falsi negativi. Formati questi insiemi è possibile ricavare i valori descritti dalle equazioni 7.1, 7.2, 7.5 e 7.6. In particolare, Precision rappresenta la probabilità di selezionare casualmente un record classificato come positivo tra tutti quelli classificati come positivi,

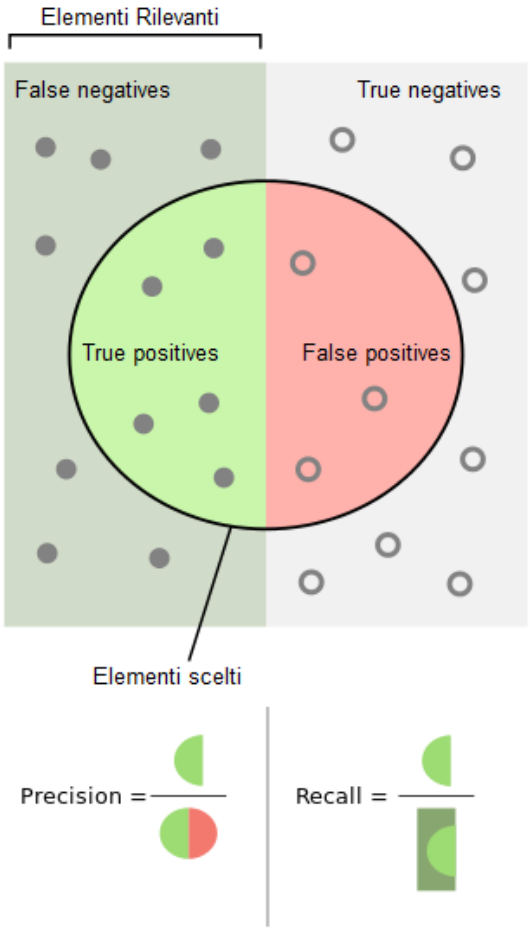

Figura 7.1: Precision e Recall

Recall indica la probabilità di selezionare casualmente un record classificato come positivo tra tutti quelli che sono realmente positivi, F-Measure combina i valori Precision e Recall e, infine, Accuracy fornisce l'informazione più generale relativa al rapporto tra i record classificati correttamente e tutti quelli presenti. I valori menzionati sono tanto migliori quanto più il loro valore è elevato (vicino a uno). Mentre la formula 7.7 invece rappresenta il Matthews correlation coefficient, il quale viene principalmente usato per classi particolarmente sbilanciate ed il suo range va da -1 a +1, la predizione perfetta la si ottiene quando esso vale 0.

7.1.2 Modello di classificazione

Una volta costruito il traning set dalla classificazione manuale descritta nel capitolo precedente, e convertito nel file in formato *arff* richiesto dal software Weka, si è passati alla creazione di un modello mediante tale software.

Weka mette a disposizione diversi algoritmi per la riduzione del numero di attributi prima della creazione del modello, ma dato il basso numero di attributi (14 canali), e per rendere più robusto possibile il classificatore, si è scelto di costruire un modello di classificazione prendendo in esame tutti e 14 i campi.

Dopo diversi test la classe di modelli più efficaci per il training set è risultata essere quella definita dagli alberi di decisione, e in particolare il modello che ha ottenuto le performance migliori è stato costruito utilizzando l'algoritmo *RandomForest* il quale verrà spiegato nel dettaglio successivamente.

Nella Tabella 7.1 si può osservare la matrice di confusione generata dal modello di classificazione così ottenuto, mentre nella Figura 7.2 viene mostrata l'interfaccia grafica del Tool, nonché l'output della costruzione del modello. Come visto

Tabella 7.1: Matrice di Confusione ottenuta da RandomForest

a	**b**	**c**	**d**	<–	classified as
123	159	46	4	|	a = 0
76	511	50	10	|	b = 1
42	122	46	8	|	c = 2
12	39	10	7	|	d = 3

nel capitolo precedente la separazione della classe 0 dalle altre classi non risulta semplice, infatti anche i risultati ottenuti da questo classificatore non possono dirsi soddisfacenti. Nel fare la valutazione della scelta del modello si è dato peso maggiore alla corretta classificazione della classe etichettata dal valore 1. Come si può vedere dalla Figura 7.2 il valore Precision, per quanto riguarda la classe di maggior interesse, risulta essere pari a 0.615, commettendo dunque un errore di quasi un 40% nella classificazione di tali momenti d'attenzione. Mentre il parametro Recall è pari a 79%, il che significa che il restante 21% delle attivazioni dovute alla concentrazione dell'autista viene perso.

Successivamente si è scelto di eliminare i record identificati dall'etichetta di classe 0, dal momento sono considerati come "corrotti" ed eliminabili in fase di registrazione.

Nella Tabella 7.2 si può osservare la matrice di confusione generata dal modello di classificazione così ottenuto, mentre nella Figura 7.3 viene mostrata l'interfaccia grafica del Tool, nonché l'output della costruzione del modello. In questo caso si sono ottenuti risultati migliori rispetto ai precedenti e come si può vedere dalla matrice il classificatore riconosce la classe 1 correttamente nell'87,63% dei casi.

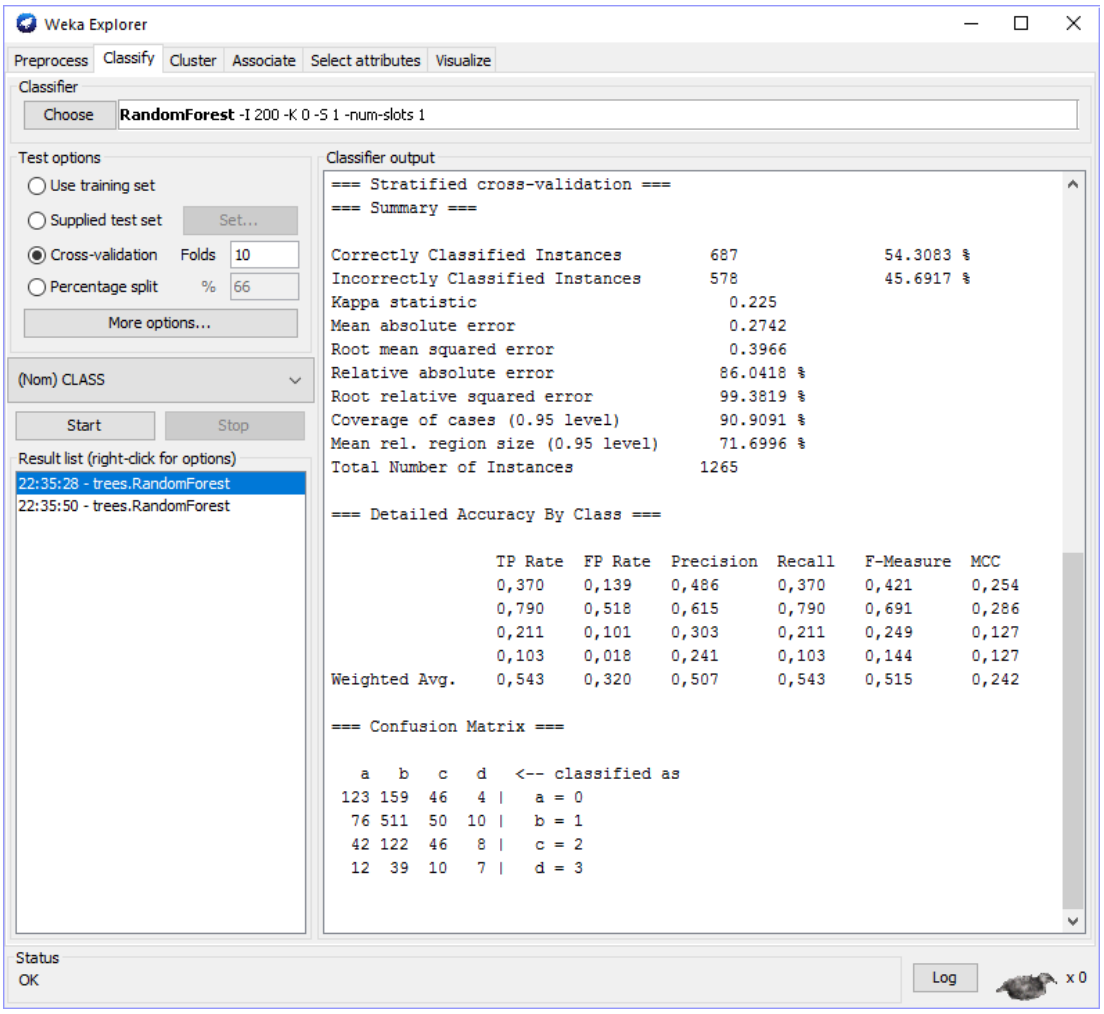

Figura 7.2: Interfaccia Weka: Modello di classificazione

Inoltre anche i valori di Precision e Recall possono considerarsi a buoni livelli essendo rispettivamente a 0,742 e a 0,876.

Tabella 7.2: Matrice di Confusione senza la Classe 0

a	**b**	**c**	<–	classified as
567	65	15	|	a = 1
149	62	7	|	b = 2
48	13	7	|	c = 3

Random Forest

Le Random Forest, sviluppate da Leo Breiman dell'Università della California, e da Adele Cutler dell'Università statale dello Utah, sono un insieme di tecniche di apprendimento ensemble per la classificazione, regressione ed altre operazioni [8]. Tali tecniche costruiscono molti alberi di decisione in fase di training e ritornano la classe rappresentante la moda delle classi (in caso di classificazione) oppure la classe predetta con maggiore probabilità (in caso di regressione) dai singoli alberi. Random Forest corregge una cattiva tendenza degli alberi di decisione classici, ossia l'overfitting sul proprio training set. L'algoritmo generale combina il metodo di boostrap aggregating (bagging), utilizzato per l'allenamento, alla selezione casuale delle feature. L'algoritmo bagging, ideato per gli alberi di decisione, funziona nel seguente modo: dato un training set $X = x_1, ..., x_n$ con risposte $Y = y_1, ..., y_n$, esso seleziona ripetutamente (B volte) un campione casuale con reinserimento dal training set ed adatta gli alberi a questi campioni:

Per $b = 1, ..., B$:

1. Campiona, con reinserimento, n esempi di training da X, Y; chiama questi X_b, Y_b.

2. Allena un albero di decisione f_b su X_b, Y_b.

Dopo l'allenamento, le predizioni per il generico campione non osservato x', possono essere fatte considerando la maggioranza dei voti (nel caso degli alberi di decisione) oppure tramite la media delle predizioni da tutti gli alberi singoli su x':

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x') \tag{7.8}$$

Tale procedura di bootstrap migliora le performance dei modelli poiché decresce la varianza senza aumentare lo sbilanciamento tra le classi (bias). Questo vuol dire che mentre le predizioni di un singolo albero sono altamente sensibili al rumore nel training set, ciò non avviene per la media di molti alberi, finché gli alberi non sono correlati. Allenare semplicemente molti alberi sullo stesso training set restituirà, come risultato, alberi fortemente correlati (oppure più volte lo stesso albero, qualora l'algoritmo di allenamento sia deterministico). Il campionamento

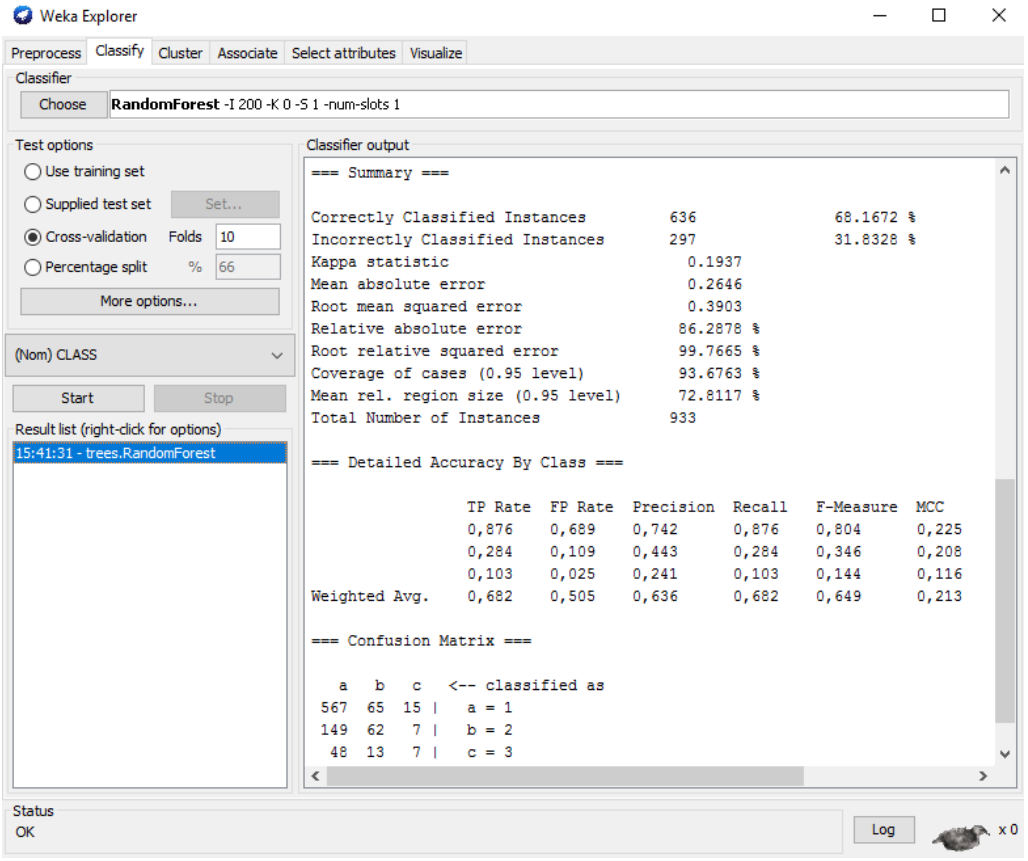

Figura 7.3: Interfaccia Weka: Modello di classificazione

effettuato da boostrap è un modo per "de-correlare" gli alberi utilizzando variazioni del training set. Il numero di alberi (e quindi di campioni), B, è un parametro libero. Generalmente lo si usa variandolo da poche centinaia a molte migliaia, a seconda della taglia e della natura del training set. Un numero ottimo di alberi può essere trovato usando, ad esempio, la cross-validation. Le foreste casuali si differenziano da questo schema generale solamente in un unico modo: utilizzano un algoritmo modificato di allenamento degli alberi che seleziona, ad ogni divisione dei candidati nel processo di apprendimento, un sottoinsieme casuale di feature, talvolta chiamato "feature bagging". La ragione per cui viene fatto ciò è la correlazione degli alberi in un ordinario campione di bootstrap: se una o più feature predicono molto bene la classe, esse verranno selezionate in molti dei B alberi, comportando alta correlazione tra di essi. Tipicamente, per un problema di classificazione con p feature, vengono utilizzate $\sqrt{p}$ feature in ogni split. Detto in modo più descrittivo, le foreste in questione selezionano casualmente degli attributi (o combinazioni di essi) ad ogni nodo per far crescere ciascun albero.

7.2 Clustering

Il clustering, o analisi dei gruppi, è un insieme di tecniche di analisi multivariata dei dati volte alla selezione e raggruppamento di elementi omogenei in un insieme di dati. Le tecniche di clustering si basano su misure relative alla somiglianza tra gli elementi. In molti approcci questa similarità, o meglio, dissimilarità, è concepita in termini di distanza in uno spazio multidimensionale. La bontà delle analisi ottenute dagli algoritmi di clustering dipende molto dalla scelta della metrica, quindi da come è calcolata la distanza. Gli algoritmi di clustering raggruppano gli elementi sulla base della loro distanza reciproca, quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme stesso.

Le tecniche di clustering si possono basare principalmente su due "filosofie":

- Bottom-Up
- Top-Down

La prima filosofia utilizza metodi aggregativi. Essa prevede che inizialmente tutti gli elementi siano considerati cluster a sé, e poi l'algoritmo provvede ad unire i cluster più vicini. L'algoritmo continua ad unire elementi fino ad ottenere un numero prefissato di cluster, oppure fino a che la distanza minima tra i cluster non supera un certo valore.

La seconda filosofia invece utilizza metodi divisivi. All'inizio tutti gli elementi sono racchiusi in un unico cluster, e poi l'algoritmo inizia a dividerli in tanti cluster di dimensioni inferiori. Il criterio che guida la divisione è naturalmente quello di ottenere gruppi sempre più omogenei. L'algoritmo procede fino a che non viene soddisfatta una regola di arresto generalmente legata al raggiungimento di un numero prefissato di cluster.

7.2.1 Clustering con Weka

Come nel caso precedente per fare elaborazioni utilizzando Weka è stato dapprima necessario convertire i dati in input nel formato *arff* richiesto dal tool. Questa volta però non sono stati utilizzati i dati classificati manualmente ma bensì tutti i record provenienti direttamente dal filtro applicato ai valori registrati dall'elettroencefalografo. Da questo dataset sono state rimossi tutti i record che contano tutti e 14 i canali al valore di 0, principalmente per ottenere maggiore efficienza in fase di Clustering, ma anche per il semplice fatto che, come si può intuire sarebbero stati tutti associati al medesimo cluster. Questa scrematura dei dati ha portato alla rimozione del 95% dei record arrivando così ad un totale di 1.560.595 record.

Anche in questo caso Weka mette a disposizione diversi algoritmi per il Clustering, seppur in numero molto minore rispetto a quelli forniti per la classificazione. In particolare quello scelto in questo caso, per motivi di efficienza e di performance, è *SimpleKMean*.

L'algoritmo K-means è un algoritmo di Clustering partizionale (filosofia Top-Down) che permette di suddividere un insieme di oggetti in K gruppi sulla base dei loro attributi. L'obiettivo che l'algoritmo si prepone è di minimizzare la varianza totale intra-cluster. Ogni cluster viene identificato mediante un centroide o punto medio. L'algoritmo segue una procedura iterativa. Inizialmente crea K partizioni e assegna ad ogni partizione i punti d'ingresso o casualmente o usando alcune informazioni euristiche. Quindi calcola il centroide di ogni gruppo. Costruisce così una nuova partizione associando ogni punto d'ingresso al cluster il cui centroide è più vicino ad esso. Quindi vengono ricalcolati i centroidi per i nuovi cluster e così via, finché l'algoritmo non converge.

Utilizzando tale algoritmo sono stati identificati 4 Cluster (come 4 erano le classi identificate precedentemente), i centroidi di tali Cluster vengono riportati nella Tabella 7.3 mentre nella Tabella 7.4 vengono riportati il numero di record per ogni Cluster e il peso (espresso in percentuale) di ogni Cluster.

Osservando i risultati ottenuti in questa fase è interessante notare come un Cluster in particolare si discosti dagli altri tre. Il Cluster identificato con l'etichetta 1 (di centroide "01001101111111") mostra come in questo caso i canali attivi siano soprattutto quelli che vanno da O2 a AF4. Dunque anche da questo studio emerge che il filtro mette in luce in particolar modo i canali che mappano l'emisfero destro del cervello, ovvero quello che si occupa di visione d'insieme, percezione, elaborazione dello spazio 3D e della visione da lontano.

Tabella 7.3: Centroidi dei Cluster identificati

Channel	Cluster 0	Cluster 1	Cluster 2	Cluster 3
AF3	0	0	0	0
F7	1	1	0	0
F3	0	0	0	0
FC5	0	0	0	0
T7	0	1	0	0
P7	0	1	0	0
O1	0	0	0	0
O2	0	1	0	0
P8	0	1	0	0
T8	0	1	1	0
FC6	0	1	1	0
F4	0	1	0	0
F8	0	1	0	1
AF4	0	1	0	0

Tabella 7.4: Numero di record per Cluster

	Cluster 0	Cluster 1	Cluster 2	Cluster 3
Numero record	560.255	530.458	335.315	134.567
Peso del Cluster	36%	34%	21%	9%

Capitolo 8

Conclusioni e possibili miglioramenti

8.1 Possibili Miglioramenti

Il sistema realizzato presenta ancora diversi margini di miglioramento, nonché sviluppi futuri ed integrazioni con altri dispositivi per rendere il tutto più robusto.

Come descritto nel capitolo 6 le attivazioni dei canali identificate dall'etichetta di classe 0 possono essere ridotte se non soppresse del tutto. Questo integrando al sistema delle smartband dotate di accelerometro allo scopo di capire la posizione del polso e delle mani, inibendo dunque l'attivazione dei canali se si identifica che le mani del soggetto sono in prossimità alla zona del volto e del capo. Inoltre tali moderne band forniscono anche la lettura del battito cardiaco, il quale potrebbe essere un'utile informazione da associare ai dati provenienti dall'elettroencefalografo allo scopo di prevenire eventuali colpi di sonno ed aiutare l'identificazione di sussulti, trasalimenti o stati d'agitazione.

L'integrazione con dati provenienti dal GPS risulta essere uno step fondamentale per associare le attivazioni dei canali a determinati punti del globo. Questo permetterebbe la correlazione tra attivazione del filtro con posizioni specifiche di una mappa stradale. Come visto nell'analisi dei dati un grande numero di attivazioni è avvenuto in istanti della guida richiedenti particolare attenzione, come ad esempio immissioni su strade, rotonde, incroci e manovre delicate. La possibilità di associare tali attivazioni ad una mappa permetterebbe successivamente di dedurre se in un determinato punto ci si aspetta un'attivazione oppure no. Una volta costruito un dataset contenente tali informazioni sarebbe possibile osservare lo storico di un tracciato per dedurre in quali punti ci si aspetta un determinato livello di concentrazione, per poi allarmare il sistema nel caso venisse a mancare

tale attenzione, poiché tale condotta potrebbe essere considerata come un comportamento a rischio d'incidente.

Possono esserci ulteriori margini di miglioramento anche per quanto riguarda la struttura del filtro realizzato. Diversi sono i parametri necessari al suo funzionamento e dunque potrebbe esistere una combinazione di tali parametri che porti a dei risultati migliori. Ad esempio si potrebbe scegliere una differente dimensione della finestra temporale per il calcolo dell'energia, differenti valori per i parametri α_1 e α_2 del filtro passa alto, o ancora differenti valori per il minimo ed il massimo valore della soglia d'energia del filtro.

Altri miglioramenti risiedono nella modellazione del classificatore automatico. Diversi sono gli algoritmi messi a disposizione da Weka e diversi sono i parametri per ciascun algoritmo. Trovare l'ottimo è un problema NP-Completo, dunque il classificatore costruito con un'altra configurazione di parametri, o con l'utilizzo di un altro algoritmo, potrebbe portare a risultati migliori.

8.2 Conclusioni

Tale lavoro di tesi magistrale può essere diviso in due parti, la prima caratterizzata dallo sviluppo della parte software e dall'implementazione e testing del sistema. La seconda invece caratterizzata dal raccoglimento e analisi dei dati. Il sistema realizzato, può essere tranquillamente utilizzato per raccogliere dati in contesti ben diversi da quelli visti nell'ambito di questa tesi, aprendo le porte ad una molteplicità di possibili studi ed esperimenti sia su automezzi, sia in laboratori ed ambienti controllati, e questo grazie alla scelta di portabilità del sistema.

Dal punto di vista puramente legato ai dati ottenuti, come si è visto nei capitoli precedenti, sono emersi risultati interessanti. Si è potuto notare come il sistema reagisca attivando i filtri associati ai canali in molti di quei contesti nella quale è richiesta una particolare concentrazione da parte dell'autista. A tal proposito risulta più interessante andare a cercare tutti quegli istanti in cui ci si aspetta un certo livello di concentrazione da parte del guidatore, ma invece tale attenzione risulta essere insufficiente o inesistente. Questo, senza dubbio, risulterebbe essere un comportamento a rischio d'incidente. Rilevandolo e segnalandolo di conseguenza si ha, dunque, la possibilità di ridurre il numero di tali comportamenti. Come discusso nell'introduzione di tale tesi, questa diminuzione porterebbe altresì ad una riduzione della probabilità d'incidente e conseguentemente alla possibilità di salvare vite umane.

Appendice A

OpenCV

OpenCV (acronimo in lingua inglese di Open Source Computer Vision Library) è una libreria software multipiattaforma nell'ambito della visione artificiale in tempo reale. È un software libero originariamente sviluppato da Intel, centro di ricerca in Russia di Niznij Novgorod. Successivamente fu poi mantenuto da Willow Garage e ora da Itseez [9].

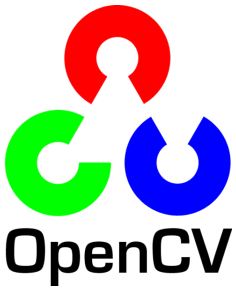

Figura A.1: Logo di OpenCV

Il linguaggio di programmazione principalmente utilizzato per sviluppare con questa libreria è il C++, ma è possibile interfacciarsi anche attraverso il C, Python e Java [9].

L'utilizzo della libreria OpenCV riveste un ruolo solamente marginale nel lavoro svolto in questa tesi, in particolare è stato utilizzato per accedere ai dati provenienti dalle videocamere, elaborare tali immagini, salvare i frame ottenuti in un file video e infine reperire tali fotogrammi allo scopo di visualizzare il video.

In particolare le classi utilizzate sono: Mat, VideoCapture e VideoWriter, le quali verranno qui brevemente descritte.

A.1 Mat

La classe Mat è una classe C++ che rappresenta la struttura dati di base di OpenCV 2.2. Ogni immagine viene rappresentate da un oggetto Mat, il quale raccoglie al suo interno una matrice contente in ogni cella un valore numerico che, a seconda della codifica scelta, rappresenta il colore associato al pixel corrispondente.

In particolare le funzioni di tale classe utilizzate sono *putText* e *resize*.

putText
: Tale funzione permette di inserire del testo all'interno di un oggetto Mat, specificandone posizione, font, colore e dimensione.

resize
: Tale funzione permette, come suggerisce il nome stesso, di ridimensionare un oggetto Mat, cambiando dunque la risoluzione dell'immagine in esso contenuto.

A.2 VideoCapture

Tale classe della libreria OpenCV permette di istanziare un oggetto che rende possibile l'acquisizione di immagini da diverse sorgenti. L'input del costruttore può essere definito da un file video o da l'id di una videocamera connessa all'elaboratore.

Una volta istanziato tale oggetto, e controllato che la sorgente sia effettivamente disponibile utilizzando la funzione *isOpened*, esso permette di ottenere fotogrammi, rappresentati da un oggetto Mat, mediante l'utilizzo della funzione *read*.

La funzione read permette di ottenere immagini da una sorgente indicata al momento della costruzione dell'oggetto VideoCapture. Nel caso la sorgente sia identificata da un file video tale funzione restituisce il prossimo frame del video. Nel caso la sorgente sia una videocamera tale funzione richiede il fotogramma corrente alla camera e resta in attesa finché essa non restituisce l'immagine, anche questa volta sotto forma di oggetto Mat. Essa è una funzione bloccante.

A.3 VideoWriter

La classe VideoWriter rappresenta un oggetto che si prende a carico di codificare e memorizzare su disco un buffer di immagini sotto forma di file video.

Tutti i parametri necessari alla costruzione del video vengono settati nel costruttore. Essi sono definiti da: directory di salvataggio, codifica di video, frame rate e risoluzione del video. La funzione *write* scrive l'immagine, identificata anche questa volta da un oggetto Mat, nel buffer del video, il quale a blocchi viene codificato e salvato nel percorso identificato nel costruttore. Infine una volta che la scrittura del video risulta completa si passa alla chiusura del file mediante la funzione *close* di VideoWriter.

Appendice B

Weka

Weka, acronimo di "Waikato Environment for Knowledge Analysis", è un software per l'apprendimento automatico sviluppato nell'università di Waikato in Nuova Zelanda. È open source e viene distribuito con licenza GNU General Public License [10].

Weka è un ambiente software interamente scritto in Java. Un semplice metodo per utilizzare questo software consiste nell'applicare dei metodi di apprendimento automatici (learning methods) ad un set di dati (dataset), e analizzarne il risultato. È possibile attraverso questi metodi, avere quindi una previsione dei nuovi comportamenti dei dati.

L'interfaccia grafica di Weka è composta da:

- Simple CLI: l'interfaccia dalla riga di comando;
- Explorer: ambiente che consente di esplorare i dati attraverso i comandi Weka:
- Preprocess permette di caricare i dati da una base dati o da un CSV e di applicare dei filtri ai dati;
- Classify applica algoritmi di classificazione e regressione;
- Cluster permette di usare tecniche di clustering;
- Associate cerca di estrarre delle Regole di associazione;
- Select attributes esegue degli algoritmi che permettono di valutare gli attributi in base alla loro utilità per la classificazione;
- Visualize visualizza un Grafico di dispersione;
- Experimenter: compie test statistici fra i diversi algoritmi di data mining;
- Knowledge Flow.

Tabella B.1: Esempio di Matrice di Confusione

a	**b**	**c**	<–	classified as
567	65	15	|	a = 1
149	62	7	|	b = 2
48	13	7	|	c = 3

B.1 Dataset

Insieme di valori e attributi presenti all'interno di una relazione. In una tabella di un database relazionale le istanze corrispondono alle righe e gli attributi alle colonne. Il formato utilizzato in Weka per la lettura dei dataset è l'ARFF(Attribute Relationship File Format), è simile al più famoso CSV (Comma-separated values) ed è equivalente alla tabella di un database relazionale.

B.2 Matrice di confusione

Le matrici di confusione vengono utilizzate per la valutazione dei classificatori utilizzati in Weka (si può vedere un esempio di matrice nella Tabella B.1).

Le *colonne* della matrice rappresentano le istanze che sono state classificate come appartenenti a quella classe. Le *righe* della matrice di confusione rappresentano le reali istanze che appartengono a quella classe.

Attraverso questo meccanismo la matrice è in grado di fornire il numero di casi che sono stati classificati correttamente e il numero di casi classificati in modo scorretto. La matrice "perfetta" viene identificata da una *matrice diagonale*, la quale si verifica quando soltanto nella diagonale sono presenti valori diversi da zero, e in questo caso ogni istanza viene classificata alla classe corretta.

Appendice C

Principali funzioni

Nella presente appendice vengono mostrate nel dettaglio le più importanti funzioni sviluppate.

C.1 Funzioni di registrazione Video

```
int recordCostantFPSVideo(int camera_id,
                          bool* stop_recording,
                          const VideoParams& params)
{

VideoCapture inputVideo(camera_id);
if (!inputVideo.isOpened())
    return -1;

char video_name[36];
outputName(video_name, camera_id);
cout << "Recording " << video_name << endl;

string path = params.output_path + video_name;
VideoWriter record_video(path, CV_FOURCC('M', 'P', '4', '2'),
                         params.fps, Size(params.video_width,
                         params.video_height));

Mat cur_frame, last_frame;
char window_name[5];
window_name[0] = 'C'; window_name[1] = 'a'; window_name[2] = 'm';
window_name[3] = camera_id + '0'; window_name[4] = 0;

double delta_time = 1000 / params.fps; // time between two frames
double elapsed_millisecs;

// ********** START RECORDING **********
clock_t current_frame_time, last_frame_time, start_time;
char c = 0;
```

```
int current_slot = 0, next_slot = 0;

// Read first frame
getFrame(inputVideo, &last_frame, params);

// Write the first frame
record_video.write(last_frame);
current_slot++;

// Initialize time
start_time = clock();
last_frame_time = start_time;

//c = waitKey(delta_time);
waitKey(delta_time);

while (!*stop_recording)
{
    // Get the frame with timestamp
    getFrame(inputVideo, &cur_frame, params);
    if (cur_frame.empty())
        break;

    // Evaluate current frame time
    current_frame_time = clock();
    elapsed_millisecs = double(current_frame_time - start_time) /
                        CLOCKS_PER_SEC * 1000;
    // If the video is too long end video
    if (elapsed_millisecs >= MAX_TIME)
        break;

    double gap_time = double(current_frame_time - last_frame_time) /
                      CLOCKS_PER_SEC * 1000;
    next_slot = currentFrameSlot(start_time,
                                 current_frame_time,
                                 delta_time);
    last_frame_time = current_frame_time;

    // *********Add the frame at the video*********
    if (next_slot == current_slot)
    {
        // Add the current frame
        record_video.write(cur_frame);
        current_slot++;

        // Wait for get the next frame
        int time_to_wait = current_slot * delta_time;
        time_to_wait -= elapsed_millisecs;
        time_to_wait = time_to_wait > 0 ? time_to_wait : 1;
        c = waitKey(time_to_wait);
    }
```

```
    else if (next_slot > current_slot) // too slow
    {
        // Add the previous frame mulple times
        for (; current_slot < next_slot; current_slot++)
        record_video.write(last_frame);

        // Add the current frame
        record_video.write(cur_frame);
        current_slot++;

        // Wait for get the next frame
        int time_to_wait = current_slot * delta_time;
        time_to_wait -= elapsed_millisecs;
        time_to_wait = time_to_wait > 0 ? time_to_wait : 1;
        c = waitKey(time_to_wait);
    }
    last_frame = cur_frame;
}

// Close the video file
record_video.release();
cout << video_name << " Saved!" << endl;
return 0;
}
```

Listing C.1: Registrazione video con FPS costante

```
int startRegistration(bool* stop_recording)
{
  VideoParams params;
  readXML(&params);

  while (true)
  {
    if (!*stop_recording)
    {
      // Start a thread for each cam
      vector<thread> threads;
      for (int i = 0; i < params.camera_number; ++i)
         threads.push_back(thread(recordVideo, i, stop_recording));


      // Wait the end of each thread for close the program
      for (int i = 0; i < params.camera_number; ++i)
         threads[i].join();
    }
  }
  return 0;
}
```

Listing C.2: Inizia le registrazioni assegnando un thread ad ogni telecamera

```
void getFrame(VideoCapture& inputVideo,
              Mat* frame,
              const VideoParams& params)
{
  int lineType = 8;
  char time[30];

  // Read the frame
  inputVideo.read(*frame);

  if (!frame->empty())
  {

    // Flip the frame
    flip(*frame, *frame, 1);

    // ********** Add time at the frame **********
    Rect subregion(0, frame->rows - 20, 300, 20);
    Mat roi(*frame, subregion);
    roi = 0;
    currentTimeMillis(time);
    putText(*frame, time, Point(0, frame->rows - 5),
            CV_FONT_HERSHEY_COMPLEX_SMALL, 1, Scalar(255, 255, 255));

    resize(*frame, *frame,
           Size(params.video_width, params.video_height));
  }
}
```

Listing C.3: Funzione che cattura ed elabora il frame corrente

```
int currentFrameSlot(clock_t start_time,
                     clock_t current_time,
                     double delta_time)
{
  double elapsed_time = double(current_time - start_time) /
              CLOCKS_PER_SEC * 1000;
  int slot = int(elapsed_time / delta_time);
  return slot;
}
```

Listing C.4: Funzione di identificazione dello slot d'inserimento del corrente frame

C.2 Filtro dati Brain Scanner

Di seguito viene riportata la funzione che si prende a carico l'accesso ai dati provenienti dal Brain Scanner, nonché la loro elaborazione e filtraggio.

```
int recordBrainData()
{
if (EE_EngineConnect() != EDK_OK)
  throw std::exception("Connessione a Emotiv Engine fallita.");

// Need this to get access to data
DataHandle hData = EE_DataCreate();
// This make the buffer (dimension = frequency * buffer_seconds)
EE_DataSetBufferSizeInSec(buffer_seconds);

// ******************** START RECORDING ********************
// j is the index of the element to replace in order to calculate
//the mean and s to calculate the threshold
int j = 0, s = 0;
// Loop until the scanner is on
while (true)
{
  state = EE_EngineGetNextEvent(eEvent);
  checkState(state);

  // Check if the scanner is on
  GetLocalTime(&current_time);
  // If scanner is off
  if (timeDifference(last_sample_time, current_time) > 1000)
  {
    if (isBrainSannerOn)
    {
      int result = sendSignal(false);
      while (result < 0)
      result = sendSignal(false);

      std::cout << "Scanner OFF" << std::endl;

      // Stop recording
      break;
    }
    isBrainSannerOn = false;
  }
  else
  {
    if (!isBrainSannerOn)
    {
      int result = sendSignal(true);
      while (result < 0)
      result = sendSignal(true);
```

```
      std::cout << "Scanner ON" << std::endl;
      std::cout << "Registrazione in corso" << std::endl;
      printf("Output file: %s\n", output_path);

    }
    isBrainSannerOn = true;
  }

  // Read data
  if (readytocollect)
  {
    EE_DataUpdateHandle(userID, hData);

    unsigned int num_samples = 0;
    EE_DataGetNumberOfSample(hData, &num_samples);

    if (num_samples != 0)
    {
      double* data = new double[num_samples];
      // This loop is necessary for read each sample in the buffer.
      for (int sample_id = 0; sample_id < num_samples; sample_id++)
      {
        fprintf(output_file, "\n");

        // Get current time
        GetLocalTime(&current_time);
        fprintf(output_file, "%02d:%02d:%02d:%03d;",
                current_time.wHour, current_time.wMinute,
                current_time.wSecond,
                current_time.wMilliseconds);
        // Update last sample time
        last_sample_time = current_time;

        // Get timestamp
        EE_DataGet(hData, ED_TIMESTAMP, data, num_samples);
        // Write the data with comma as decimal separator
        char converted_data[30];
        outputConverter(data[sample_id], converted_data);
        fprintf(output_file, "%s;", converted_data);

        // Loop for each channel
        int num_of_channels = sizeof(targetChannelList) /
                              sizeof(EE_DataChannel_t);
        for (int i = 0; i < num_of_channels; i++)
        {
          EE_DataGet(hData, targetChannelList[i], data, num_samples);

          if (j == WINDOWS_LENGTH)
            j = 0;

          tmp_average[i] = tmp_average[i] - buffer_average[i][j] +
```

```
                    data[sample_id];
        buffer_average[i][j] = data[sample_id];

        if (start_counter >= WINDOWS_LENGTH)
        {
          average[i] = tmp_average[i] / WINDOWS_LENGTH;
          norm = data[sample_id] - average[i];
        }
        else
        {
          average[i] = tmp_average[i] / start_counter;
          norm = data[sample_id] - average[i];
        }

        // High Pass filter
        highpass[i] = ALPHA * (highpass[i] +  norm -
                      previous_value[i]);
        previous_value[i] = norm;

        // Compute the energy
        energy[i] = energy[i] - buffer_energy[i][j] +
                    highpass[i] * highpass[i];
        buffer_energy[i][j] = highpass[i] * highpass[i];

        // *********** COMPUTE THE THRESHOLD ***********
        if (s == THRESHOLD_LENGTH)
          s = 0;

        tmp_threshold[i] = tmp_threshold[i] -
                           buffer_threshold[i][s] + energy[i];
        buffer_threshold[i][s] = energy[i];

        if (start_counter >= THRESHOLD_LENGTH)
          threshold[i] = tmp_threshold[i] / THRESHOLD_LENGTH;
        else
          threshold[i] = tmp_threshold[i] / start_counter;

        // Compute min and max threshold
        if (energy[i] > max_energy[i])
        {
          max_energy[i] = energy[i];
          current_min_threshold[i] = max_energy[i] *
                                     MIN_THRESHOLD_PERCENT;
          current_max_threshold[i] = max_energy[i] *
                                     MAX_THRESHOLD_PERCENT;

          // The min threshold set at the beginning could drop!
          if (current_min_threshold[i] < MIN_THRESHOLD)
          {
            current_min_threshold[i] = MIN_THRESHOLD;
            current_max_threshold[i] = MAX_THRESHOLD;
```

```
          }
        }

        // Check if threshold is more than minimun threshold
        if (threshold[i] < current_min_threshold[i])
        threshold[i] = current_min_threshold[i];

        // Check if threshold is less than maximun threshold
        if (threshold[i] > current_max_threshold[i])
        threshold[i] = current_max_threshold[i];

        // ************ END THRESHOLD COMPUTATION *************

        // Print the output (file of console)
        if (flag_file_print)
        {
          if (energy[i] > threshold[i])
            fprintf(output_file, "1;");
          else
            fprintf(output_file, "0;");
        }
        else
        {
          printf("%f;%d;%f;%f;%f;%f;", highpass[i],
                 start_counter, energy[i], threshold[i],
                 current_min_threshold[i],
                 current_max_threshold[i]);
          if (energy[i] > threshold[i])
            printf("%d;", 1);
          else
            printf("%d;", 0);
        }
      }
      start_counter++;
      s++;
      j++;

      // Print the pure data in the output file
      if (flag_file_print)
      {
        for (int i = 0; i < num_of_channels; i++)
        {
          EE_DataGet(hData, targetChannelList[i], data,
                     num_samples);
          outputConverter(data[sample_id], converted_data);
          fprintf(output_file, "%s;", converted_data);
        }
      }
    }
    delete[] data;
  }
```

```
  }
  // Wait the next sample
  Sleep(8);
}

// Free the buffer
EE_DataFree(hData);
EE_EngineDisconnect();
EE_EmoStateFree(eState);
EE_EmoEngineEventFree(eEvent);

fclose(output_file);

// Compress the output file
cout << "Compressing the file..." << endl;
fileZipper(output_name, output_path);

// Delete the file uncompressed
remove(output_path);

return 0;
}
```

Listing C.5: Accesso e filtraggio dei dati EEG

C.3 Codice VideoPlayer

In questa sezione viene mostrato il codice che prende a carico la riproduzione del video dal secondo prescelto nell'applicazione VideoPlayer.

```
int play(string video_name, int seconds)
{
  VideoCapture inputVideo(video_name);
  if (!inputVideo.isOpened())
  return -1;

  double fps = inputVideo.get(CV_CAP_PROP_FPS);
  int total_frame = (int)inputVideo.get(CV_CAP_PROP_FRAME_COUNT);
  double delta_time = 1000 / fps; // time between two frames

  // Start the video at the selected point
  int frame_to_seek = fps * seconds;
  if (frame_to_seek < total_frame)
  seekTo(inputVideo, fps * seconds, video_name);
  else
  seekTo(inputVideo, total_frame - 1, video_name);

  Mat cur_frame;
  bool paused = true;
```

```
  // Press ESC to stop the recording
  while (inputVideo.isOpened())
  {
    // Check if the video is over
    int current_frame_number = inputVideo.get(CV_CAP_PROP_POS_FRAMES);
    if (current_frame_number == total_frame)
    paused = true;

    // Check if the video is paused or not
    if (!paused)
    {
      inputVideo >> cur_frame;
      if (cur_frame.empty())
      break;

      // Show the frame
      imshow(video_name, cur_frame);
    }

    // Wait instruction
    int c = waitKey(delta_time);
    switch (c)
    {
      case(27): // Esc button is pressed
      return 1;

      case('b'): // rewind a second
      seekTo(inputVideo, -fps, video_name);
      break;

      case('n'): // forward a second
      seekTo(inputVideo, fps, video_name);
      break;

      case('c'): // rewind a frame
      seekTo(inputVideo, -1, video_name);
      break;

      case('v'): // forward a frame
      seekTo(inputVideo, 1, video_name);
      break;

      case(' '): // Paused the video
      paused = !paused;
      break;
    }
  }
  return 1;
}
```

Listing C.6: Riproduzione del video dal secondo desiderato

Bibliografia

[1] Piramide di Heinrich: fantasia o realtà? http://marziomarigo.postilla.it/2016/02/15/piramide-di-heinrich-fantasia-o-realta/

[2] Elettroencefalografo, http://it.wikipedia.org/wiki/Elettroencefalografo

[3] EEG - elettroencefalogramma, http://www.neurologiapsichiatria.it/home/esami-strumentali/eeg-elettroencefalogramma/

[4] Epoc, www.emotiv.com/epoc.php

[5] Insight, www.emotiv.com/insight.php

[6] High-pass, http://en.wikipedia.org/wiki/High-pass

[7] Emisfero destro e sinistro: Come funzionano cervello e creatività, http://www.lematpercorsi.com/2012/11/emisfero-destro-e-sinistro-come-funzionano-cervello-e-creativita/

[8] Random forest, https://en.wikipedia.org/wiki/Random_forest

[9] OpenCV, http://opencv.org/

[10] Weka, http://www.cs.waikato.ac.nz/ml/weka/

[11] Classification and Clustering, https://www.ibm.com/developerworks/library/os-weka2/

[12] Ilya Kuzovkin (2013) Adaptive Interactive Learning: a Novel Approach to Training Brain-Computer Interface Systems

Ringraziamenti

Giunto alla fine di questa tesi desidero ringraziare il professor Nicola Zingirian, nonché relatore di questa tesi magistrale, il quale mi fornito utili consigli nello svolgimento di tale progetto, e mi ha permesso di realizzare questo innovativo ed ambizioso progetto.

Un particolare ringraziamento va anche al signor Lazzarini Moreno, il quale, durante il suo lavoro, si è prestato alla raccolta dati essenziale per il lavoro di ricerca di tale tesi.

Vorrei inoltre ringraziare la mia famiglia, perché mi è sempre stata accanto e non mi ha fatto mai mancare il suo sostegno ed il suo aiuto durante tutti questi anni.

Grazie a Daniela, la quale ha sempre sostenuto le mie scelte e mi ha motivato nei momenti più bui. La sua presenza, disponibilità, ma sopratutto pazienza è stata provvidenziale in questi anni.

Infine, ma non meno importante, vorrei ringraziare Michele Mattiazzi, il quale, sebbene circondato da mille impegni, ha trovato il modo di aiutarmi in momenti difficili, e a consigliarmi in momenti di dubbio.

Ancora una volta grazie a tutti.