Università degli Studi di Salerno

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Implementazione e Ottimizzazione di Algoritmi per l'analisi dei Dati Oculari in Pazienti con Parkinson.

Relatori
Prof. Andrea Francesco Abate
Dott.ssa Lucia Cascone

Candidato
Luigi Miranda
Matricola: 0512110723

Anno Accademico 2022/2023

Abstract

Il presente studio propone l'elaborazione statistica delle informazioni acquisite da un dispositivo per il tracciamento oculare, il Tobii Pro Glasses 3, al fine di individuare dati utili per la diagnosi precoce del morbo di Parkinson. Oggetto dell'analisi è l'insieme dei dati provenienti da un esperimento condotto dal reparto di Neurologia dell'ospedale San Giovanni di Dio e Ruggi d'Aragona di Salerno, al quale ha partecipato un gruppo misto di soggetti affetti da morbo di Parkinson e soggetti sani.

Questa tesi si basa sullo studio degli algoritmi utilizzati per identificare le così dette *fissazioni*(movimenti oculari). Il lavoro ha coinvolto un processo di ottimizzazione degli algoritmi esistenti, evidenziando le differenze chiave tra di essi al fine di migliorare l'accuratezza nella rilevazione di questi movimenti oculari. È stato sviluppato un codice scalabile che consente una maggiore flessibilità nell'applicazione di tali algoritmi per diverse configurazioni.

Un passo ulteriore è stato compiuto attraverso la creazione di un processo di automatizzazione nella raccolta dei dati che semplifica notevolmente la fase di acquisizione degli stessi e consente agli utenti di concentrarsi esclusivamente sull'analisi dei risultati.

Infine è stata condotta un'analisi statistica approfondita dei dati che ha permesso di identificare pattern significativi nelle fissazioni dei pazienti con Parkinson, contribuendo a una comprensione puntuale e dettagliata dei loro movimenti oculari.

L'analisi descritta nella presente tesi va approfondita con ulteriori test e confronti, ma rappresenta un buon punto di partenza per studi futuri.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Il morbo di Parkinson è una patologia neurodegenerativa cronica che colpisce il sistema nervoso centrale, provocando la graduale degenerazione delle cellule nervose nella regione del cervello responsabile del controllo dei movimenti volontari. Caratterizzata principalmente da tremori, rigidità e difficoltà di coordinazione motoria, questa malattia ha un impatto significativo sulla qualità della vita dei pazienti [1]. La diagnosi precoce risulta fondamentale per intraprendere interventi terapeutici tempestivi e migliorare la qualità della vita dei pazienti. In questa prospettiva, l'eye tracking(tracciamento oculare) emerge come una tecnologia cruciale per l'analisi dei dati oculari, fornendo preziosi *insight* nella comprensione delle patologie neurodegenerative [2]. Il tracciamento oculare può aiutare in diversi modi la diagnosi e il monitoraggio di questa malattia, in particolare attraverso:

- Sintomi oculari: alcuni pazienti con Parkinson manifestano sintomi specifici legati ai movimenti oculari, come la difficoltà a muovere gli occhi o il rallentamento dei movimenti saccadici (rapidissimi movimenti degli occhi da un punto all'altro). L'eye tracking può registrare e quantificare queste anomalie [3].
- Diagnosi precoce: identificare il Parkinson nelle sue fasi iniziali è cruciale per l'efficacia del trattamento. Le alterazioni nei movimenti oculari potrebbero emergere prima di altri sintomi motori evidenti, rendendo l'eye tracking uno strumento potenzialmente utile per una diagnosi precoce [4].
- Monitoraggio del progresso della malattia: oltre alla diagnosi, il tracciamento oculare potrebbe essere utilizzato per monitorare la progressione della malattia, valutando come i movimenti oculari cambiano nel tempo.
- Valutazione dell'efficacia del trattamento: l'eye tracking può essere utilizzato per determinare l'efficacia di nuovi trattamenti o modifiche al regime terapeutico, osservando eventuali miglioramenti o deterioramenti nella funzione oculare [5].
- Studio delle manifestazioni non motorie: poiché il Parkinson può influenzare vari aspetti della funzione cerebrale, l'eye tracking potrebbe aiutare a studiare altre manifestazioni non motorie, come problemi di memoria o attenzione, valutando come i pazienti seguono visivamente gli stimoli [6].

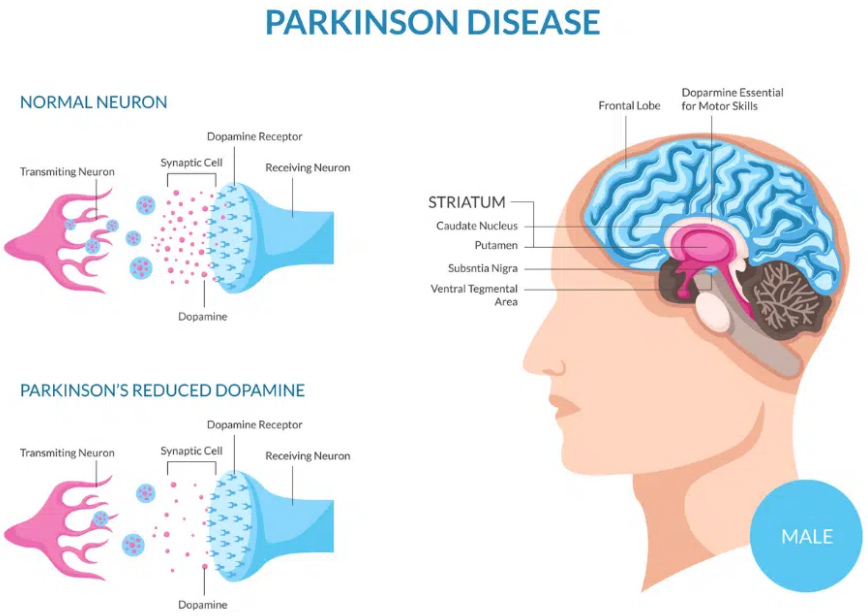

Figura 1.1: Il cervello e il Parkinson

L'eye tracking, quindi, consente la registrazione e l'analisi dei movimenti oculari, rivelando informazioni importanti su parametri chiave come pupille, saccadi, blink e fissazioni.

Il presente elaborato si concentra sulla fase di fissazione - breve periodo in cui gli occhi rimangono fermi su un punto specifico - consentendo al cervello di elaborare informazioni dettagliate in quella regione. Lo studio delle fissazioni oculari, attraverso l'utilizzo dell'eye tracking, consente di analizzare la focalizzazione dell'attenzione visiva degli individui su determinati stimoli. Comprendere pattern comuni sui risultati dello studio può risultare fondamentale in molteplici settori come nel contesto delle patologie neurodegenerative tra cui il morbo di Parkinson.

In particolare, gli aspetti su cui si concentra questo lavoro sono: la raccolta automatizzata dei dati, l'ottimizzazione del codice già esistente e l'approfondimento degli algoritmi di rilevazione delle fissazioni.

Una parte fondamentale di questo lavoro è stata dedicata all'automatizzazione della raccolta dei dati. Lo sviluppo di un sistema efficiente e robusto consente di gestire grandi quantità di dati per un ampio campione di utenti, semplificando la fase preliminare di ricerca, la lettura e l'elaborazione dei dati. Il sistema è stato progettato per garantire la massima portabilità e scalabilità, infatti può essere utilizzato su Windows, MacOS e Linux e non necessita di interventi complessi sul codice. Queste caratteristiche consentono agli utenti di poter lavorare direttamente sull'oggetto della loro ricerca senza preoccuparsi degli aspetti tecnici e implementativi relativi all'acquisizione dei dati.

Nei capitoli successivi, viene esplorato in dettaglio il concetto di fissazioni, analizzando il loro ruolo nel contesto delle patologie neurodegenerative come il morbo di Parkinson. Viene presentata un'approfondita panoramica degli algoritmi esistenti per la rilevazione delle fissa-

zioni, delineando le diverse metodologie utilizzate per interpretare i dati oculari. Per i diversi algoritmi studiati, sono stati analizzati più parametri in modo da comprendere come ognuno di essi influenzi i risultati. L'implementazione di tali algoritmi, offre una panoramica chiara delle potenzialità e delle limitazioni di ciascun approccio. Sono state poi esplorate le ragioni dietro le scelte implementative, sottolineando l'importanza di adattare le tecniche esistenti per ottenere risultati accurati e significativi nella specifica cornice di analisi dei dati oculari legati al Parkinson.

Infine è stata effettuata un'analisi statistica sulle fissazioni per individuare pattern frequenti nei pazienti con Parkinson. Nello specifico, è stato posto un particolare accento sulla creazione di file organizzati contenenti i risultati utili per lo studio. Questa organizzazione mira a fornire una visione completa e dettagliata, sia da una prospettiva più specifica che da una più ampia. Infatti, per ciascun utente coinvolto nello studio, sono stati creati due file:

- il primo contenente informazioni sugli stimoli visualizzati e le fissazioni da essi generate; questo fornisce una panoramica completa del comportamento visivo di ogni individuo, consentendo analisi più approfondite e personalizzate.

- il secondo contenente i risultati complessivi, specificamente: numero di fissazioni, media della durata, time to first fixation ecc.

Invece per una visione aggregata dei dati, sono state create statistiche mensili che consentono una valutazione del comportamento oculare in un periodo specifico. Tale organizzazione temporale agevola la rilevazione di trend e variazioni nel tempo. Al fine di fornire una visione complessiva e immediata relativa all'intero periodo di studio, è stato creato un file globale che raccoglie le statistiche di tutti i mesi analizzati.

In conclusione, il presente lavoro di tesi rappresenta un valido strumento di supporto per approfondire lo studio delle fissazioni e, più in generale, delle altre tematiche connesse all'eye tracking.

Capitolo 2

Eye Tracking e fissazioni per la diagnosi del Parkinson

2.1 Tracciamento oculare

Il tracciamento oculare, in inglese eye tracking, è una tecnologia avanzata che consente di monitorare e registrare i movimenti degli occhi di un individuo mentre osserva un determinato ambiente, dispositivo o contenuto. Questa tecnologia si basa sulla registrazione della posizione e del movimento degli occhi, consentendo di analizzare il punto esatto in cui una persona sta guardando in un dato momento [7]. Il principale obiettivo dell'eye tracking è quello di comprendere e analizzare il comportamento visivo degli individui al fine di ottenere informazioni preziose su come interagiscono con stimoli visivi specifici.

Nel contesto medico [8], e in particolare nello studio del morbo di Parkinson, l'eye tracking assume un ruolo cruciale nella scoperta di nuovi metodi per diagnosticare il morbo precocemente; la sua importanza risiede nella capacità di fornire dati obiettivi sulla performance visiva dei pazienti. Attraverso questa tecnologia, è possibile osservare e registrare eventuali alterazioni nei movimenti saccadici e nelle transizioni tra punti di fissazione, fenomeni spesso influenzati dalla progressione della malattia. Nel processo diagnostico, l'eye tracking può contribuire a identificare precocemente i segni distintivi associati al morbo di Parkinson, fornendo ai medici un valido ausilio per la formulazione di diagnosi più tempestive ed accurate [9].

Sono molteplici, infatti, gli studi che riguardano l'analisi dei dati registrati da questi strumenti per individuare proprietà che possano aiutare nella diagnostica del Parkinson. Di seguito ne sono riportati i principali con le relative conclusioni:

1. *«Fixation Duration and Pupil Size as Diagnostic Tools in Parkinson's Disease»*:

   Obiettivi: analizzare come varia la stabilità e la dimensione della pupilla in condizione di luce ottimale, nonché il confronto tra i movimenti oculari effettuati da malati di Parkinson (PD) e persone sane (HC) durante la fissazione duratura.

   Metodi: lo studio ha coinvolto 43 persone sane (37% maschi) e 50 pazienti PD (66% maschi) con malattia unilaterale da lieve a moderata. I dati sono stati registrati attra-

verso un eye tracker screen-based chiamato Tobii Pro Spectrum.
Infine, l'analisi dei dati è stata effettuata tramite l'utilizzo della regressione logistica.

Risultati: i pazienti sani sono in grado di mantenere le pupille stabili su un oggetto fermo per un periodo di tempo più lungo rispetto ai malati PD. Inoltre, in condizioni di luce ottimale le pupille dei malati PD mostrano un diametro più piccolo rispetto alla controparte sana dello studio [10].

2. *«Oculomotor Performances Are Associated With Motor and Non-motor Symptoms in Parkinson's Disease»*:

   Obiettivi: valutare una probabile relazione tra deficit nei movimenti oculari e i sintomi del morbo di Parkinson.

   Metodi: allo studio hanno partecipato 37 malati di Parkinson e 39 persone sane. I movimenti oculari sono stati registrati attraverso un eye tracker video-based, di nome EyeLink 1000, capace di tracciare la pupilla e la riflessione corneale attraverso un sistema ad infrarossi.

   Risultati: i pazienti malati presentano una minore stabilità delle pupille durante le fissazioni, una latenza saccadica più lunga, una maggiore differenza tra la velocità oculare e la velocità dell'oggetto di interesse durante i movimenti oculari di inseguimento, minor numero di saccadi e un campo visivo più ristretto durante la visione libera.
   Inoltre, è stato osservato che le alterazioni riscrontate nei pazienti malati sono significativamente associate a diverse variabili cliniche correlate a sintomi motori e non motori, tra cui la durata della PD, UPDRS III, punteggio di equilibrio di Berg e punteggio RBD [11].

3. *«Pupil light reflex dynamics in Parkinson's disease»*

   Obiettivi: indagare su una probabile relazione tra la disfunzione autonomica (o disautonomia) e il riflesso pupillare alla luce difettoso (PLR) nei malati di Parkinson. Sappiamo, infatti, che la PLR è un fattore associato alla presenza di disautonomia (malfunzionamento del sistema nervoso) che è uno dei principali sintomi non-motori del morbo.

   Metodi: lo studio ha coinvolto 50 pazienti malati di Parkinson e 43 persone sane. La posizione degli occhi e le dimensioni della pupilla sono state registrate con un tracker screen-based, Tobii Pro Spectrum, e sono stati utilizzati due stimoli luminosi: uno breve (100ms) e uno più lungo (1000ms). Come indicatore della disautonomia è stata utilizzata la presenza di ipotensione ortostatica (OH).
   Infine, il confronto tra i dati appartenti ai diversi gruppi è stato effettuato tramite il modello lineare ad effetti misti e i test statistici non parametrici.

   Risultati: la velocità di picco di costrizione (un parametro per la PLR) è più lenta nei pazienti PD, e questa decresce ulteriormente nel sottogruppo di pazienti PD con OH,

rispetto alle persone sane.

Inoltre, anche l'ampiezza e la velocità di dilatazione delle pupille in risposta agli stimoli luminosi risultano minori nei pazienti PD, e in particolare nei pazienti PD con OH, in confronto ai pazienti sani [12].

2.2 Le fissazioni

Le fissazioni, nel contesto della percezione visiva, rappresentano pause prolungate e concentrate su specifiche aree di interesse nell'ambito del campo visivo. Durante queste pause, l'occhio rimane relativamente stabile, consentendo una maggiore elaborazione delle informazioni visive in quella particolare regione. La misurazione della stabilità della fissazione può essere utile per valutare l'attenzione visiva e la capacità di mantenere lo sguardo su un determinato bersaglio [13]. La misurazione delle fissazioni può essere utilizzata in vari contesti, come la ricerca scientifica, la valutazione delle abilità visive e l'ottimizzazione dell'interfaccia uomo-macchina. Le tecnologie di tracciamento oculare consentono di registrare e analizzare le caratteristiche delle fissazioni per ottenere informazioni sulla selezione visiva, l'attenzione, la percezione e il coinvolgimento cognitivo durante le attività visive [14].

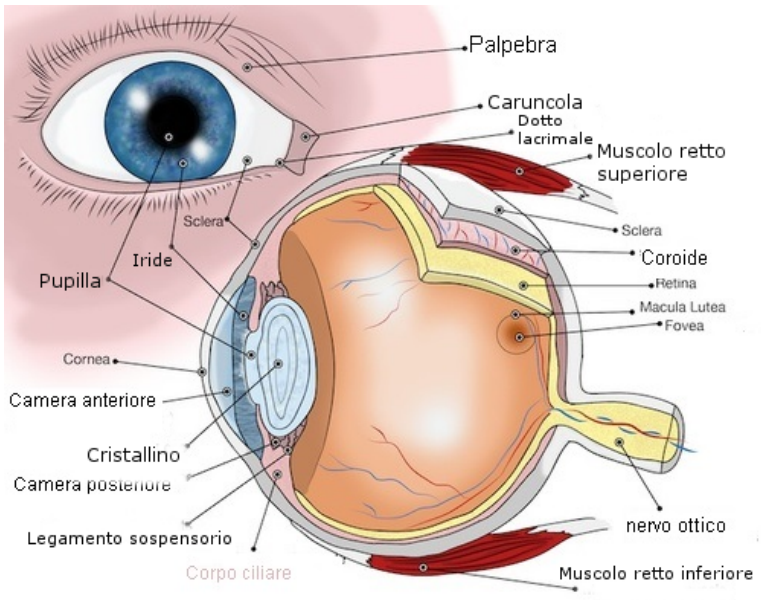

Figura 2.1: Anatomia dell'occhio umano (Fonte: Percezione Visiva)

2.2.1 Durata della fissazione

La durata della fissazione rappresenta il periodo di tempo in cui gli occhi rimangono fermi su un punto specifico. In generale, durante la lettura, la durata media della fissazione si aggira intorno ai 200-300 millisecondi. Tuttavia, questa durata può variare in base alla difficoltà del testo. Parole complesse o poco familiari possono richiedere fissazioni più lunghe, a volte superiori a 350 millisecondi, mentre parole brevi o familiari potrebbero richiedere solo 150-200 millisecondi. Nell'osservazione di immagini o scene, la durata delle fissazioni varia in

modo ancora più ampio. In una scena familiare o con pochi dettagli, una fissazione può durare circa 150 millisecondi. Tuttavia, in presenza di dettagli nuovi o interessanti, la durata della fissazione può estendersi fino a 600 millisecondi o più, a seconda della complessità e dell'interesse del soggetto [15].

2.2.2 Stabilità della fissazione

La stabilità della fissazione indica quanto stabili sono gli occhi durante una fissazione. In un individuo con una buona stabilità della fissazione, gli occhi sono in grado di mantenere la posizione su un punto di interesse entro un range di 0,5 a 1 minuto d'arco per la maggior parte del tempo di fissazione. Tuttavia, piccoli movimenti oculari, come i tremoli, drift e micro-saccadi, possono naturalmente verificarsi anche durante una "fissazione stabile". I tremoli sono piccoli movimenti oscillatori ad alta frequenza, tipicamente nell'ordine di 30-100 Hz, con un'ampiezza di meno di 0,5 minuti d'arco. I drift sono movimenti più lenti che possono causare deviazioni dell'ordine di 1-5 minuti d'arco ogni secondo. Le micro-saccadi, d'altro canto, sono movimenti veloci simili alle saccadi, ma molto più piccoli, tipicamente nell'ordine di pochi minuti d'arco. La presenza di questi movimenti durante una fissazione è normale e aiuta a prevenire la sbiadimento retinico. Tuttavia, un eccesso di questi movimenti o movimenti fuori dai range tipici possono indicare una fissazione instabile. Ad esempio, se gli occhi si muovono ripetutamente oltre 5-10 minuti d'arco durante una fissazione, ciò potrebbe essere considerato instabile e potenzialmente problematico per attività come la lettura [16].

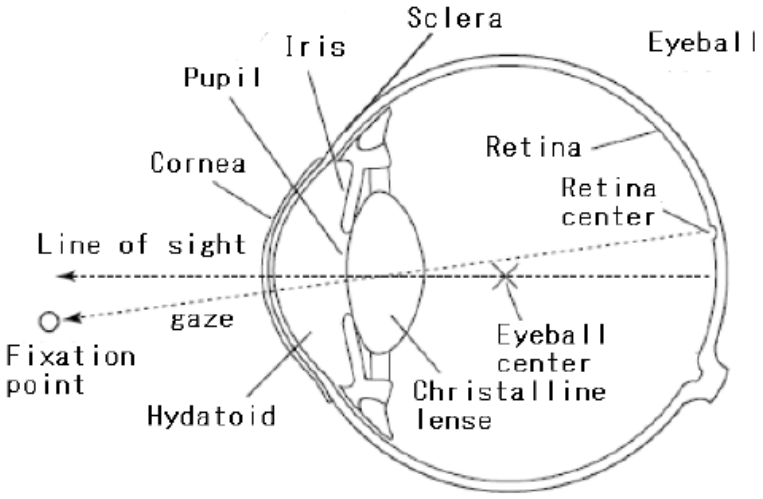

Figura 2.2: Sguardo e linea visiva di un punto di fissazione

2.2.3 Precisione della fissazione

La precisione della fissazione si riferisce alla capacità di mantenere lo sguardo su un bersaglio specifico con precisione. Una fissazione precisa implica che gli occhi siano allineati con precisione sul punto di interesse e non deviano involontariamente verso altre direzioni. In una fissazione considerata "precisa", la posizione del punto di fissazione rimane entro un range di 0,5 minuti d'arco dal bersaglio desiderato per la maggior parte del tempo [15].

2.2.4 Soglia di fusione

La soglia di fusione è la minima differenza di posizione tra due punti che gli occhi possono percepire come un'unica immagine durante una fissazione. Per la maggior parte degli individui, la soglia di fusione si situa in un range tra 30 e 60 secondi d'arco. Ciò significa che se le immagini retiniche provenienti dai due occhi divergono di un valore inferiore a questa soglia, il cervello può fondere le due immagini in una sola immagine percettiva. Valori superiori a questa soglia possono portare a percezioni doppie o confuse. Una soglia di fusione più bassa indica una maggiore capacità di mantenere gli occhi allineati su un punto e di fondere le informazioni visive [15].

2.3 Il progetto Parkinson

Il progetto Parkinson è un progetto nato dalla collaborazione tra l'Università degli studi di Salerno e l'ospedale San Giovanni di Dio e Ruggi d'Aragona a Salerno.
Lo studio coinvolge un totale di 45 persone, ma solo 17 (9 maschi e 8 femmine) sono risultati idonei all'analisi e di questi 13 sono malati PD e 4 sono individui sani (3 con miopia). L'obiettivo di questo progetto è quello di analizzare i dati registrati dall'eye tracker Tobii Pro Glasses 3 e tentare di individuare delle caratteristiche utili alla diagnosi precoce del morbo di Parkinson.

2.3.1 Tobii Pro Glasses 3

Un eye tracker è un dispositivo in grado di registrare i movimenti oculari concentrandosi, in particolare, su dove, cosa e per quanto tempo il soggetto guarda uno specifico punto della scena visiva. La tecnica di tracciamento più utilizzata è la riflessione corneale del centro della pupilla (PCCR), questa consiste nel porre una fonte luminosa in prossimità dell'infrarosso, la luce che viene riflessa in modo evidente nella cornea e nella pupilla del soggetto viene fotografata. A partire da questa immagine, poi, vengono determinati lo spostamento e la direzione dello sguardo.
Esistono tre categorie principali di eye trackers:

- Screen-based (o remoti): sono dispositivi montati vicino ad un schermo e richiedono che il soggetto sia seduto davanti ad esso, senza uscire dal range di tracciamento dell'eye tracker, chiamato headbox. Sono indicati per test in laboratorio o in altri piccoli contesti.

- VR eye trackers: sono sensori di tracciamento oculare integrati all'interno di visori VR. Anche se il display è posizionato troppo vicino agli occhi per poter registrare la convergenza degli occhi, è possibile comunque ottenere un buon tracciamento dello sguardo andando a combinare le informazioni sulla profondità degli oggetti virtuali. Principalmente sono utilizzati per studiare il comportamento e l'attenzione visiva del soggetto all'interno della realtà virtuale.

- Mobili (o glasses): sono dispositivi indossabili che permettono di analizzare i movimenti oculari in molteplici contesti e senza alcun limite di libertà di movimento. Nonostante

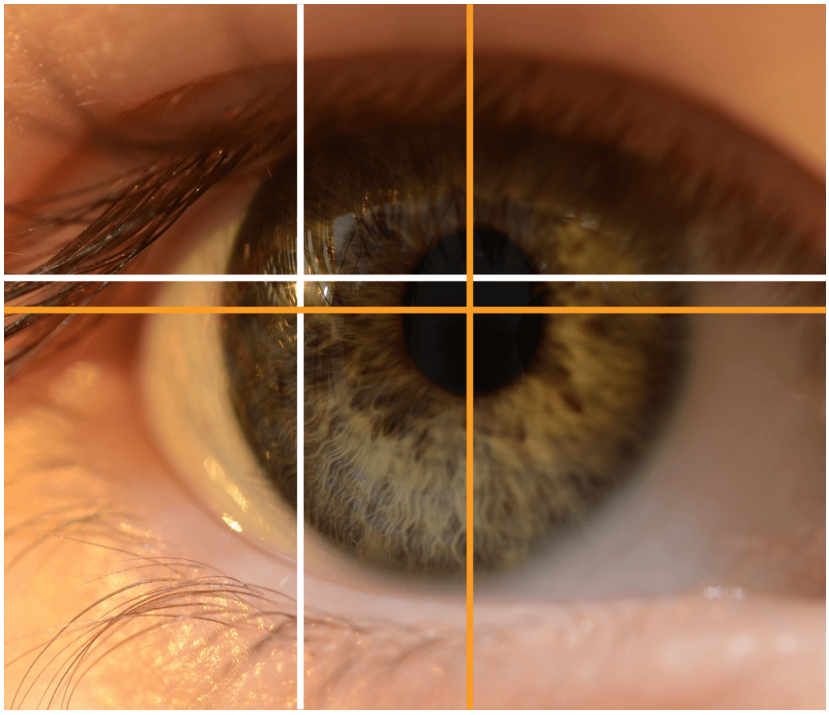

Figura 2.3: PCCR (Link Fonte).

siano montati sulle montature di occhiali, i dati registrati sono precisi e non risentono dei movimenti della testa perché la telecamera di tracciamento è bloccata sulle coordinate della testa [17, 18].

Il dispositivo Tobii Pro Glasses 3 rientra nella categoria degli eye trackers mobili. È in grado non solo di registrare i movimenti oculari ma anche l'audio dell'ambiente circostante fornendo uno strumento utile per comprendere il comportamento e il processo decisionale del soggetto. È costituito da un'unità principale indossabile provvista di molteplici sensori e videocamere, e da un'unità esterna di registrazione che si collega all'altra via cavo.

L'unità principale è progettata per essere comoda e leggera in modo da non influenzare i dati registrati ed è composta da molteplici elementi:

1. Illuminatori ad infrarosso: sono 8 per ciascun occhio e sono utili per illuminare gli occhi senza bisogno di fonti luminose esterne.

2. Cavo di collegamento: collega l'unità principale all'unità di registrazione in modo da memorizzare i dati al suo interno.

3. Telecamera di scena: cattura un video Full HD di ciò che si trova davanti al soggetto. Ha un campo visivo di 160° in diagonale, 95° in orizzontale e 63° in verticale.

4. Microfono: registra i rumori prodotti dal soggetto e dall'ambiente che lo circonda.

5. Telecamere di tracciamento: sono 4 (2 per ciascun occhio) e registrano i movimenti e l'orientamento degli occhi.

Figura 2.4: Unità principale (Link Fonte).

6. Attacco per accessori: utile per l'aggancio di accessori, come ad esempio le lenti di protezione.

7. Nasello: addizionale e presente in varie misure.

Una batteria Li-on ricaricabile e sostituibile fornisce energia all'unità principale e all'unità di registrazione. Quest'ultima è un piccolo computer che registra e memorizza i dati di tracciamento oculare, i rumori e i video ripresi dalla telecamera di scena all'interno di una scheda SD rimovibile. Le funzionalità sono controllabili attraverso un'applicazione Tobii Pro Glasses 3 dedicata. È composta dai seguenti elementi:

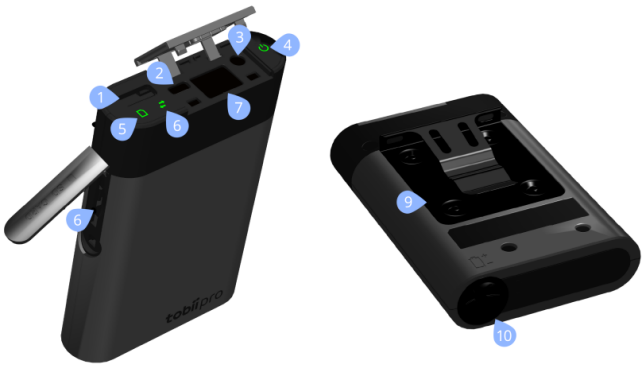

Figura 2.5: Unità di registrazione (Link Fonte).

1. Connettore per l'unità principale: collegare l'unità di registrazione all'unità principale.

2. Connettore micro-USB: utile per collegare l'unità di registratore ad un caricatore USB e fornire energia alla batteria.

3. Connettore da 3,5mm: consente la sincronizzazione dei dati con sorgenti esterne.
4. Pulsante e indicatore LED di accensione: il pulsante viene utilizzato per accendere o spegnere l'unità, mentre l'indicatore segnala lo stato della batteria e la sua capacità residua;
5. LED di stato per la scheda SD: indica se la scheda SD è presente e se i dati vengono correttamente salvati al suo interno.
6. LED di stato della connessione: indica se l'unità di registrazione è collegata all'unità principale.
7. Porta LAN: connette l'unità di registrazione al dispositivo dove è presente l'applicazione di controllo attraverso un cavo Ethernet.
8. Slot per scheda SD: spazio dove inserire la scheda SD.
9. Clip: gancio per legare l'unità alla cintura del soggetto.
10. Compartimento batteria: spazio adibito alla batteria Li-on.

A supporto degli sviluppatori viene fornito anche un software Tobii Pro Glasses 3 Controller che consente di utilizzare i dati raccolti e di integrarli in altre applicazioni. L'interfaccia grafica user-friendly di questo software permette agevolemente di avviare, arrestare e riprodurre le registrazioni effettuate, ma anche di modificare le impostazioni del dispositivo come la frequenza di campionamento (50Hz o 100Hz), l'esposizione della camera, lo zoom, il tipo di segnale di sincronizzazione e la risoluzione video.
È possibile, inoltre, esportare i dati memorizzati in file usufruibili anche in modalità offline, come i file in formato CSV. Grazie alle molteplici API di Tobii Pro Glasses 3 Controller, poi, è possibile inserire questi dati all'interno di applicazioni personalizzate e di sincronizzarli con altre sorgenti di dati [19].

2.3.2 Dati di partenza dello studio: Descrizione Task e immagini

Durante la procedura di raccolta dati ciascun partecipante ha indossato l'unità principale del dispositivo Tobii Pro Glasses 3 ed è stato posto davanti ad un monitor in condizioni di oscurità. Successivamente, gli è stata data piena libertà per esplorare i scenari visivi mostrati a video senza alcun preavviso sui loro contenuti. L'esperimento ha avuto una durata media di circa 6 minuti con la visualizzazione di immagini appartenenti a 4 task differenti ed è stato articolato nei seguenti step:

1. Fase di calibrazione:
    - Visione schermo bianco: 2 secondi.
    - Calibrazione: fissazione di un target (figura 2.8) che appare per 4 secondi su ciascun angolo dello schermo.
    - Visione schermo bianco: 2 secondi.

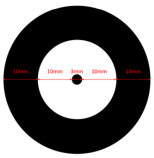

Figura 2.6: Target di calibrazione (Link Fonte).

2. Task n.1: Esplorazione visiva di 4 scenari complessi (ciascuna immagine è intervallata da una croce di fissazione):

    - Visione croce nera fissa: 2 secondi.
    - Visione immagini: 4 secondi.

Figura 2.7: Esempio immagine task n.1.

3. Task n.2: Esplorazione visiva di 4 ambienti percorribili (ciascuna immagine è intervallata da una croce di fissazione):

    - Visione croce nera fissa: 2 secondi.
    - Visione immagini: 4 secondi.

4. Task n.3: Disautonomia (ciascuna delle 4 immagini scelte randomicamente da un set di 7 immagini è intervallata da una croce di fissazione):

    - Visione croce nera fissa: 2 secondi.
    - Visione immagini: 4 secondi.

5. Task n.4: Decision making implicito (ciascuna delle 16 immagini è intervallata da una croce di fissazione):

    - Visione croce nera fissa: 2 secondi.

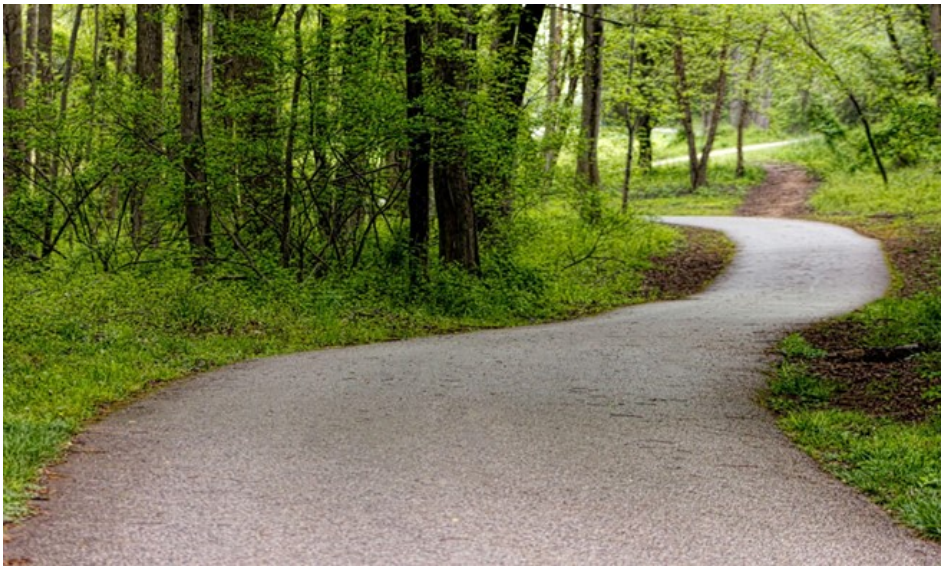

Figura 2.8: Esempio immagine task n.2.

Figura 2.9: Esempio immagine task n.3.

- Visione immagini: 4 secondi.

Successivamente, vengono ripetuti gli step 2-5 ma le immagini di ogni task vengono visualizzate in modo randomico ed almeno una croce di fissazione è lampeggiante.

Infine, è necessario sottolineare che non è stato utilizzato alcun tipo di supporto per stabilizzare la testa dei partecipanti, ai quali è stato comunque richiesto di restare il più immobile possibile in modo da non generare errori di misurazione.

Al termine di ciascuna registrazione, Tobii Pro Glasses 3 crea all'interno della root principale una sottocartella il cui nome corrisponde all'oraio UTC dell'inizio registrazione e alla data di acquisizione. Tuttavia, questo nome può essere modificato sia durante la registrazione che in seguito. All'interno di questa sottocartella vengono creati diversi elementi, tra cui:

Figura 2.10: Esempio immagine task n.4.

- Cartella meta: contiene metadati addizionali che vengono inseriti nella cartella all'avvio della registrazione,
  - RuVersion: contiene la versione del firmware dell'unità all'avvio della registrazione.
  - RuSerial: contiene il numero di serie dell'unità di registrazione.
  - HuSerial: contiene il numero di serie dell'unità principale.
- Eventdata.gz: contiene i dati relativi alle azioni dall'utente verso il dispositivo o viceversa, come ad esempio muovere, scuotere o inclinare gli occhiali. Al suo interno contiene diversi parametri:
  - type: indica il tipo dell'oggetto JSON, in questo caso "syncport";
  - timestamp: riporta l'istante di tempo preciso (in secondi) in cui sono stati registrati i dati e ha come riferimento il tempio di avvio del video;
  - data: oggetto JSON che definisce il contenuto di ogni acquisizione;
  - direction: specifica la direzione dell'evento;
  - value: indica il valore della tensione generata dal segnale TTL, può essere 1 (alto) oppure 0 (basso);
- Gazedata.gz: contiente tutte le informazioni riguardo lo sguardo, come la posizione e la dilatazione delle pupille. In presenza di blink (chiusura momentanea di entrambi gli occchi) vi è assenza di dati.

  Ha molteplici elementi:
  - type: specifica il tipo dell'oggetto JSON, in questo caso "gaze";
  - timestamp: indica l'istante di tempo preciso (in secondi) in cui sono stati acquisiti i dati e ha come riferimento il tempio di inizio del video;

```
{"type":"syncport","timestamp":1.593044,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":2.093044,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":2.593044,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":10.100332,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":11.092886,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":20.100484,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":21.100453,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":30.131729,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":31.131729,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":40.162712,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":41.162712,"data":{"direction":"out","value":0}}
{"type":"syncport","timestamp":50.193693,"data":{"direction":"out","value":1}}
{"type":"syncport","timestamp":51.193693,"data":{"direction":"out","value":0}}
```


Figura 2.11: Esempio di file eventdata.gz.

  - gaze2d: rappresenta la posizione corrispondente allo sguardo in un determinato istante attraverso coordinate normalizzate in un intervallo compreso tra 0 e 1. La prima coordinata si riferisce alla posizione sull'asse delle ascisse, la seconda su quella delle ordinate. In particolare, la coordinata (0, 0) corrisponde all'angolo in alto a sinistra dello schermo, la coordinata (1, 1) all'angolo in basso a destra;
  - gaze3d: indica la posizione di convergenza del vettore dello sguardo in relazione alla telecamera di scena. Si basa su un sistema a tre dimensioni (x, y, z) dove: x si riferisce al movimento orizzontale rispetto alla telecamera di scena, y al movimento verticale e z si riferisce alla distanza tra il punto che si sta osservando e la telecamera di scena. Tutti i valori espressi in millimetri;
  - eyeleft/eyeright: è un oggetto JSON che contiene i dati relativi ai singoli occhi;
  - gazeorigin: indica la posizione dell'occhio rispetto alla telecamera di scena sfruttando un sistema a tre dimensioni (x, y, z), dove: x rappresenta la posizione orizzontale dell'occhio, y la sua posizione verticale e z indica la distanza tra l'occhio e la telecamera di scena. Tutti i valori sono riportati in millimetri;
  - gazedirection: rappresenta la direzione dello sguardo attraverso un sistema a tre dimensioni (x, y, z), dove: x rappresenta la posizione orizzontale dell'occhio rispetto al punto che si sta osservando, y la sua posizione verticale rispetto a ciò che si osserva e z indica la distanza tra l'occhio e il punto che si sta osservando. Tutti i valori sono espressi in millimetri;
  - pupildiameter: specifica il diametro della pupilla in millimetri;

- Imudata.gz: contiene informazioni provenienti da tre Unità di Misura Inerziale (IMU):
  - type: indica il tipo di oggetto JSON, in questo caso "imu";
  - timestamp: rappresenta l'istante di tempo preciso (in secondi) in cui sono stati tracciati i dati e ha come riferimento il tempio di inizio del video;
  - data: rappresenta un oggetto JSON con al suo interno i dati relativi ad ogni acquisizione;

```
{
  "type":"gaze",
  "timestamp":10.21,
  "data":{
    "gaze2d":[0.468,0.483],
    "gaze3d":[37.543,-18.034,821.265],
    "eyeleft":{
      "gazeorigin":[28.799,-7.165,-23.945],
      "gazedirection":[0.0463,-0.0337,0.998],
      "pupildiameter":2.633},
    "eyeright":{
      "gazeorigin":[-28.367,-5.353,-21.426],
      "gazedirection":[0.0436,0.00611,0.999],
      "pupildiameter":2.782}
  }
}
```


Figura 2.12: Esempio di file gazedata.gz.

- accelerometer: indica l'accelerazione dell'unità principale tramite un sistema a tre coordinate (x, y, z). Quando questa è ferma, il valore della coordinata z è -9,8 m/s2. Viene campionato a 100HZ;
- gyroscope: indica la ruotazione dell'unità principale lungo i tre assi x, y, z. I dati sono campionati a 100Hz ed espressi in °/sec;
- magnometer: rappresenta l'intensità del campo magnetico attorno all'unità principale sempre attraverso un sistema di tre coordinate (x, y, z) e può essere utilizzato per determinare l'orientamento dell'unità. Viene campionato a 10Hz e i dati sono espressi in micro Tesla;


```
{
  "type":"imu",
  "timestamp":1.33,
  "data":{
    "accelerometer":[-0.0427,-0.920,0.472],
    "gyroscope":[2.601,0.0822,-0.179]
  }
}
```


Figura 2.13: Esempio di file imudata.gz.

- Recording.g3: descrive la registrazione e il suo stato attuale. Contiene molteplici parametri:
  - uuid: valore che identifica univocamente la registrazione;
  - name: nome della registrazione;
  - versione: versione del file.g3;
  - duration: durata della registrazione espressa in secondi;
  - created: data e tempo UTC dell'avvio della registrazione;
  - timezone: timezone selezionata all'inizio della registrazione;

- snapshot: oggetto JSON dove sono memorizzati il nome e l'istante di tempo in cui è stata catturata un'immagine dalla telecamera di scena;
- cameracalibration: oggetto JSON che ha al suo interno i dati relativi alla telecamera di scena:
  * position: posizione 3D della telecamera di scena al momento della calibrazione;
  * rotation: rotazione del dispositivo al momento della calibrazione. Viene espressa tramite una matrice 3x3;
  * focal-lenght: lunghezza focale 2D utilizzata per la registrazione;
  * skew: inclinazione della telecamera di scena;
  * principal-point: punto principale 2D della telecamera di scena;
  * radial-distortion: distorsione radiale 3D della telecamera di scena;
  * tangential-distortion: distorsione tangenziale 2D della telecamera di scena;
  * resolution: risoluzione del video registrato dalla telecamera di scena. Viene espressa in pixel;
- scenecamera: oggetto JSON che contiene le informazioni relative alla telecamera di scena;
- eyecameras: oggetto JSON che ha al suo interno il nome del file dove è memorizzato il video registrato dalle telecamere in prossimità degli occhi;
- gaze: oggetto JSON che contiene le informazioni relative allo sguardo;
- imu: oggetto JSON che ha al suo interno il nome del file imu.gz corrispondente;
- events: oggetto JSON che contiene il nome del file con al suo interno i dati relativi agli eventi che si verificano durante la registrazione corrente;
- meta-folder: nome della cartella dove sono memorizzati i metadati corrispondenti [19];

Nel progetto Parkinson vengono presi in considerazione solo alcuni dei dati registrati dal dispositivo Tobii Pro Glasses 3:

- i dati relativi alla posizione dello sguardo (gaze2d, presente nel file gazedata.gz);
- i dati relativi al diametro della pupilla destra e sinistra con il loro corrispondente timestamp (pupildiameter, presente nel file gazedata.gz);
- il video registrato dalla videocamera di scena (scenevideo);

In particolare, sono stati ignorati principalmente i dati presenti nel file imudata.gz (accelerometer, gyroscope, magnometer) i quali, influenzati da oggetti esterni, possono dar vita ad errori nella misurazione dell'orientamento del dispositivo.

```
{"uuid":"912558d3-bd01-4cc7-9645-5862bb1ab8f9",
 "name":"user022.19012023.162651",
 "meta-folder":"meta","version":2,
 "duration":398.42958276800005,
 "created":"2019-02-14T11:51:25.758300Z",
 "timezone":"Europe/Rome",
 "scenecamera":{
                "file":"scenevideo.mp4",
                "snapshots":[{
                            "file":"snap0.jpg",
                            "time":1.7519777680000002}],
                            "camera-calibration":{
                                        "position":[0.0,0.0,0.0],
                                        "focal-length":[919.237060546875,918.60406494140625],
                                        "rotation":[[-0.9999392032623291,0.010468284599483013,0.0034661802928894758],
                                                    [-0.010423211380839348,-0.99986404180526733,0.012776084244251251],
                                                    [0.0035994527861475945,0.012739178724586964,0.99991238117218018]],
                                        "skew":0.0,"principal-point":[952.3115234375,521.01629638671875],
                                        "radial-distortion":[-0.062516331672668457,0.09869636595249176,-0.061660025268793106],
                                        "tangential-distortion":[1.8019341951003298e-05,-2.5275587177020498e-05],
                                        "resolution":[1920,1080]},
                                        "gaze-overlay":false},
                                        "eyecameras":null,
                                        "gaze":{
                                                "file":"gazedata.gz",
                                                "samples":39851,
                                                "valid-samples":31444},
                                                "events":{
                                                        "file":"eventdata.gz"},
                                                               "imu":{"file":"imudata.gz"}}
```


Figura 2.14: Esempio di file recording.g3.

Capitolo 3

Gli algoritmi di identificazione delle fissazioni

Negli ultimi dieci anni, il tracciamento oculare ha guadagnato popolarità come finestra privilegiata per esplorare i processi visivi e cognitivi degli osservatori. Gli studi condotti in vari ambiti, come la scansione di immagini, la guida, l'aritmetica, l'analogia e la lettura, hanno sfruttato il tracciamento oculare per analizzare il comportamento degli individui. In queste ricerche, l'analisi dei movimenti oculari si concentra comunemente su fissazioni (pause su regioni di interesse informative) e saccadi (movimenti rapidi tra le fissazioni). Parametri di analisi comuni includono la durata delle fissazioni o dello sguardo, le velocità saccadiche, le ampiezze saccadiche e vari parametri basati su transizioni tra fissazioni e/o regioni di interesse.

Algoritmi di identificazione mal definiti possono produrre troppe o troppe poche fissazioni, o essere eccessivamente sensibili e creare dei dati anomali, influenzando quindi l'interpretazione dei risultati. Karsh e Breitenbach [20] hanno dimostrato in modo rigoroso come algoritmi di identificazione diversi possano produrre interpretazioni drasticamente diverse anche quando si analizzano protocolli identici. Pertanto, buoni algoritmi di identificazione garantiscono posizioni e durate valide per fissazioni e saccadi, influenzando a loro volta le inferenze, come la complessità di elaborazione e i percorsi di ricerca visiva.

Il presente capitolo si colloca all'interno del contesto degli studi di Salvucci e Goldberg, i quali hanno svolto un ruolo di grande importanza nella classificazione degli algoritmi di identificazione delle fissazioni e nello sviluppo dei loro criteri di implementazione [21].

3.1 Tassonomia degli algoritmi

La tassonomia qui presentata ha l'obiettivo di classificare gli algoritmi di identificazione delle fissazioni in base alle loro caratteristiche spaziali e temporali, come riassunto nell'immagine 3.1. Nel processo di elaborazione di questa tassonomia, è stato individuato un set essenziale di criteri in grado di catturare al meglio le differenze tra gli algoritmi comuni esistenti. Sebbene si sarebbe potuta proporre una tassonomia più complessa, capace di contemplare distinzioni più sottili e/o algoritmi ibridi, la tassonomia di base proposta fornisce una caratterizzazione utile dei principali tipi di algoritmi di identificazione. Questa tassonomia costituirà un punto

di partenza fondamentale per il confronto e la valutazione degli algoritmi di identificazione esistenti, argomento che esploreremo nelle sezioni successive.

Per quanto concerne le caratteristiche spaziali, la tassonomia individua tre criteri distintivi che delineano altrettanti tipi principali di algoritmi: quelli basati sulla velocità, sulla dispersione e sull'area. Gli algoritmi basati sulla velocità enfatizzano le informazioni di velocità

Criteria		Representative Algorithms				
		I-VT	I-HMM	I-DT	I-MST	I-AOI
Spatial	Velocity-based	X	X			
	Dispersion-based			X	X	
	Area-based					X
Temporal	Duration sensitive			X		X
	Locally adaptive		X	X	X	

Figura 3.1: Tassonomia degli algoritmi di identificazione

nei protocolli di eye-tracking, sfruttando la caratteristica che i punti di fissazione presentano basse velocità, mentre i punti di saccade manifestano velocità elevate. Gli algoritmi basati sulla dispersione identificano le fissazioni come gruppi di punti consecutivi entro una particolare dispersione o distanza massima. La dispersione misura quanto i punti di fissazione si discostano dalla media ovvero il punto centrale o medio della distribuzione spaziale dei punti di fissazione. Gli algoritmi basati sulle Aree di Interesse (AOI) nell'analisi dell'eye-tracking sono progettati per identificare e analizzare il comportamento degli occhi in specifiche regioni di un'immagine, di uno schermo o di un altro input visivo.

Per quanto riguarda le caratteristiche temporali, gli algoritmi possono utilizzare una soglia di tempo fissa o adattiva. Le fissazioni raramente durano meno di 100 ms e spesso si collocano nell'intervallo di 200-400 ms, quindi è possibile utilizzare una soglia temporale fissa che ci permette di distinguere le fissazioni da altri tipi di movimenti oculari. In altri contesti, invece, l'algoritmo può adattarsi dinamicamente e modificare questa soglia temporale, che non sarà più una costante ma potrà variare in base alle informazioni provenienti dai punti precedenti o circostanti. In pratica, ciò significa che l'algoritmo può essere più sensibile ai cambiamenti nel comportamento degli occhi nel corso del tempo, adattandosi alle specifiche condizioni dell'osservatore. Questa flessibilità rende l'algoritmo più robusto e adatto a una

gamma più ampia di contesti, contribuendo a migliorare l'accuratezza nell'identificazione di fissazioni e saccadi.

Consideriamo due individui, uno ha uno sguardo relativamente stabile, mentre l'altro mostra movimenti oculari più ampi e frequenti. Se si utilizzasse una soglia temporale fissa per distinguere tra fissazioni e saccadi, questa potrebbe essere troppo rigida per individuare con precisione le fissazioni nell'individuo con movimenti oculari più ampi e potrebbe essere troppo permissiva nell'individuo con uno sguardo più stabile. Con una soglia temporale dinamica, l'algoritmo può adattarsi alle caratteristiche specifiche di ciascun individuo. Inoltre, una soglia temporale dinamica può essere utile nel rilevare cambiamenti nella dinamica visiva di uno stesso individuo nel corso del tempo. Ad esempio, se una persona passa da uno sguardo stabile a movimenti oculari più ampi a causa di stanchezza o altre variabili, una soglia dinamica può adattarsi a tali cambiamenti, garantendo una migliore precisione nell'identificazione di fissazioni e saccadi.

3.2 Algoritmi rappresentativi

Al fine di consentire una comparazione rigorosa delle classi di algoritmi nella nostra tassonomia, risulta conveniente formalizzare alcuni algoritmi campione che servano a rappresentare le idee fondamentali incarnate da ciascuna classe di algoritmi. In questa sezione, descriveremo cinque algoritmi: I-VT, I-HMM, I-DT, I-MST e I-AOI, che estraggono gli aspetti cruciali di numerosi algoritmi esistenti e esprimono le loro tecniche di base nel modo più semplice possibile. La figura 3.1 illustra come questi algoritmi si collocano all'interno della tassonomia proposta. Naturalmente, esistono molti altri algoritmi possibili che potremmo includere in questa analisi, come ad esempio algoritmi ibridi con multiple euristiche; tuttavia, il seguente lavoro si è concentrato solo su alcuni di essi, in particolare sugli I-VT e I-DT.

3.2.1 Velocity-Based Algorithms

Velocity-Threshold Identification (I-VT)

L'Algoritmo Velocity-Threshold Identification (I-VT) rappresenta uno dei metodi più semplici da comprendere e implementare nell'analisi dei movimenti oculari. Si tratta di un metodo basato sulla velocità che separa i punti di fissazione e i punti di saccade in base alle loro velocità punto-punto. Si possono delineare due profili di velocità dei movimenti oculari: bassa velocità per le fissazioni (cioè, $<$100 gradi al secondo) e alta velocità (cioè, $>$300 gradi al secondo) per le saccadi [22, 23]. Questo aspetto dei movimenti oculari rende la discriminazione basata sulla velocità abbastanza diretta e robusta.

In teoria si può creare un algoritmo I-VT che adatta e modifica la soglia di velocità per ogni punto elaborato, ma dato che la velocità nei movimenti oculari durante le fissazioni segue generalmente delle regole ben definite, fortemente influenzate dalla fisiologia del sistema oculare e dalle caratteristiche fisiche del moto oculare, risulta sufficiente per il processo di identificazione delle fissazioni l'utilizzo di criteri statici (una soglia di velocità fissa). I-VT inizia calcolando le velocità punto-punto per ciascun dato grezzo. Ogni velocità è calcolata

```
I-VT (protocol, velocity threshold)

Calculate point-to-point velocities for each
point in the protocol

Label each point below velocity threshold as
a fixation point, otherwise as a saccade
point

Collapse consecutive fixation points into
fixation groups, removing saccade points

Map each fixation group to a fixation at the
centroid of its points

Return fixations
```

Figura 3.2: Pseudocodice dell'algoritmo I-VT.

come la distanza tra il punto corrente e il punto successivo (o precedente). I-VT classifica quindi ciascun dato come un punto di fissazione o di saccade sulla base di una semplice soglia di velocità: se la velocità del dato è al di sotto della soglia, abbiamo un punto di fissazione, altrimenti abbiamo un punto di saccade.

Il processo, quindi, raggruppa i punti di fissazione consecutivi in gruppi di fissazione e scarta i punti di saccade. Infine, I-VT traduce ciascun gruppo di fissazione in una tupla $< x, y, t, d >$ utilizzando il centroide (cioè, centro di massa) dei punti x e y, il tempo del primo punto come t e la durata dei punti come d. Lo pseudocodice per il metodo I-VT è mostrato nell'immagine 3.2.

I-VT richiede la specifica di un parametro, la soglia di velocità. Se le velocità angolari possono essere calcolate (cioè, la distanza tra l'occhio e gli stimoli visivi è nota), la soglia di velocità punto-punto può essere approssimata da una ragionevole soglia di velocità angolare. Tuttavia, quando sono note solo le velocità punto-punto, potrebbe essere necessario dedurre un valore appropriato per la soglia di velocità basandosi su aspetti della raccolta dati (ad esempio, frequenza di campionamento) insieme a un'analisi esplorativa dei dati. Nel caso del Tobii Pro Glasses 3 la soglia di velocità è impostata a $\frac{30°}{\text{sec}}$; inoltre viene assegnata una durata minima di fissazione di 60ms.

HMM Identification (I-HMM)

I-HMM utilizza un modello di Markov nascosto, un tipo di modello probabilistico che descrive come un sistema (in questo caso, i movimenti oculari) evolve nel tempo in modo probabilistico [24]. L'HMM utilizzato in I-HMM ha due stati che rappresentano rispettivamente la distribuzione di velocità per i punti di saccade e i punti di fissazione [25]. I due stati dell'HMM contengono una probabilità di osservazione, ovvero la distribuzione di velocità prevista in ciascuno stato e una probabilità di transizione che indica la probabilità di rimanere nello stesso stato o passare a un altro stato. L'algoritmo è implementato con una programmazione

```
I-HMM (protocol, HMM)

Calculate point-to-point velocities for each
point in the protocol

Decode velocities with two-state HMM to
identify points as fixation or saccade points

Collapse consecutive fixation points into
fixation groups, removing saccade points

Map each fixation group to a fixation at the
centroid of its points

Return fixations
```

Figura 3.3: Pseudocodice dell'algoritmo I-HMM.

dinamica. In pratica, cerca di determinare quale sequenza di stati (saccade o fissazione) è più probabile data la sequenza di velocità osservata. Dopo aver associato ciascun punto del dataset a uno stato, raggruppa i punti di fissazione consecutivi in gruppi e restituisce le fissazioni come i centroidi di questi gruppi. I parametri di I-HMM comprendono le probabilità di osservazione e di transizione nei due stati HMM: due parametri di probabilità di osservazione (la media e la varianza della distribuzione) e due probabilità di transizione per ciascuno stato, per un totale di otto parametri. Sebbene abbiamo più parametri rispetto ad I-VT, i parametri di I-HMM possono essere appresi attraverso una procedura chiamata reestimation [26].

In sostanza, I-HMM mira a fornire un'identificazione più robusta delle fissazioni rispetto ai metodi basati su soglia fissa, sfruttando un approccio probabilistico e un modello di Markov nascosto.

3.2.2 Dispersion-Based Algorithms

Dispersion-Threshold Identification (I-DT)

L'approccio di I-DT si basa sul fatto che i punti di fissazione, a causa della loro bassa velocità, tendono a raggrupparsi strettamente. Questo algoritmo identifica le fissazioni come gruppi di punti consecutivi che non superano una soglia specifica di dispersione o distanza massima. Poiché le fissazioni hanno in genere una durata di almeno 100 ms, gli algoritmi basati sulla dispersione spesso incorporano una soglia di durata minima di 100-200 ms [27] per contribuire ad attenuare la variabilità delle apparecchiature.

L'algoritmo I-DT si basa sull'algoritmo di riduzione dei dati di Widdel [27]. Utilizza una finestra scorrevole che si sposta tra i punti consecutivi, verificando la presenza di possibili fissazioni. La finestra inizia all'inizio del dataset e inizialmente incorpora un numero minimo di punti, determinato dalla soglia di durata e dalla frequenza di campionamento fornite. I-DT verifica quindi la dispersione dei punti nella finestra sommando le differenze tra i valori massimi e minimi delle coordinate x e y: Dispersione $D = [max(x) - min(x)] + [max(y) - min(y)]$

```
I-DT (protocol, dispersion threshold,
duration threshold)

While there are still points

    Initialize window over first points to
    cover the duration threshold

    If dispersion of window points <=
    threshold

        Add additional points to the window
        until dispersion > threshold

        Note a fixation at the centroid of the
        window points

        Remove window points from points

    Else

        Remove first point from points

Return fixations
```

Figura 3.4: Pseudocodice dell'algoritmo I-DT.

Se la dispersione supera la soglia, la finestra non rappresenta una fissazione e si sposta di un punto a destra creando una nuova finestra di dimensione pari sempre alla soglia di durata. Se la dispersione è inferiore alla soglia, la finestra rappresenta una fissazione. In questo caso, la finestra si espande verso destra fino a quando la dispersione supera la soglia. La finestra finale viene registrata come una fissazione al centroide dei punti della finestra, specificando tempo di inizio e di fine. Questo processo continua muovendo la finestra verso destra fino alla fine del dataset.

L'algoritmo I-DT richiede due parametri, la soglia di dispersione e la soglia di durata. La soglia di dispersione può essere stimata dall'analisi esplorativa dei dati. La soglia di durata è tipicamente impostata su un valore compreso tra 100 e 200 ms [27], a seconda delle esigenze di elaborazione dell'acquisizione dei dati.

3.2.3 Area-based Algorithms

Area-of-Interest Identification (I-AOI)

L'algoritmo Area-of-Interest Fixation Identification (I-AOI) è progettato per identificare fissazioni che si verificano solo all'interno di specifiche aree di interesse nel campo visivo. Queste aree di interesse sono rappresentate da regioni rettangolari e sono considerate unità di informazione visiva [28]. L'intero processo inizia associando i dati registrati con le aree di interesse. I punti di dati situati all'interno di una di queste aree vengono etichettati come punti di fis-

```
I-AOI (protocol, duration threshold,
target areas)

Label each point as a fixation point for the
target area in which it lies, or as a saccade
point if none

Collapse consecutive fixation points for the
same target into fixation groups, removing
saccade points

Remove fixation groups that do not span the
minimum duration threshold

Map each fixation group to a fixation at the
centroid of its points

Return fixations
```

Figura 3.5: Pseudocodice dell'algoritmo I-AOI.

sazione per quell'area specifica, mentre quelli al di fuori di tutte le aree di interesse vengono considerati come rappresentanti saccadi.

Successivamente, i punti di fissazione consecutivi all'interno della stessa area di interesse vengono raggruppati, formando così dei "gruppi di fissazione". Durante questa fase, i punti che rappresentano movimenti saccadici vengono scartati, concentrando l'attenzione solo sulle vere e proprie fissazioni. Per rendere l'analisi più significativa, l'algoritmo I-AOI utilizza un parametro di durata. Vengono eliminati i gruppi di fissazione che cadono al di sotto di una soglia di durata specifica. Questa operazione aiuta a distinguere le vere fissazioni dai semplici movimenti di saccade attraverso le aree di interesse.

Infine, i gruppi di fissazione rimanenti vengono trasformati in "tuple di fissazione". L'obiettivo principale di questo algoritmo è quindi concentrarsi sulle fissazioni che si verificano all'interno di specifiche aree di interesse, fornendo un quadro più dettagliato e significativo rispetto all'identificazione generica delle fissazioni.

3.2.4 Confronti e valutazioni

I-VT è semplice da implementare, è efficiente e può essere eseguito facilmente in tempo reale. Tuttavia, può risultare meno accurato quando le velocità dei punti si avvicinano alla soglia a causa di rumori dell'eye-tracker o altri errori in fase di rlevazione. In particolare, quando le velocità dei punti sono vicine alla soglia (ad esempio, a metà strada tra saccadi e fissazioni), questa rigida soglia di velocità può causare "blips" nell'identificazione, ovvero gruppi di fissazioni o saccadi con solo uno o pochi punti consecutivi. Ci sono diverse soluzioni a questo problema, come aggregare le fissazioni consecutive come singolo punto di fissazione su un determinato bersaglio, oppure stabilire durate minime per fissazioni o saccadi; ad esempio, Sen e Megaw [23] hanno stabilito che i punti di saccade rimanessero sopra una soglia di velocità per almeno 10 ms.

Method	Accuracy	Speed	Robustness	Impl. Ease	Parameters
Velocity Threshold (I-VT)	√	√√	×	√√	1
Hidden Markov Model (I-HMM)	√√	√	√√	√ / ×[a]	8 / 0 [a]
Dispersion Threshold (I-DT)	√√	√	√√	√	2
Minimum Spanning Tree (I-MST)	√	×	√√	×	2
Area-of-Interest (I-AOI)	×	√	√	√	1+[b]

Figura 3.6: Riepilogo dei metodi di identificazione.

I-DT è un algoritmo lineare, che produce risultati di identificazione più robusti, evitando il problema dei "blip" grazie alla soglia di durata minima. Il principale svantaggio di I-DT è l'uso di due parametri altamente interdipendenti; ad esempio, una piccola soglia di dispersione e una grande soglia di durata potrebbe non identificare alcuna fissazione. Pertanto, I-DT richiede un'accurata impostazione dei parametri ma fornisce buoni risultati con un'implementazione abbastanza semplice.

IVT è più semplice e computazionalmente più efficiente, ma può soffrire di problemi di precisione legati alla soglia di velocità. D'altra parte, IDT è più robusto grazie all'uso della dispersione spaziale, ma può richiedere maggiori risorse computazionali. La scelta tra i due metodi dipende dalle esigenze specifiche dell'analisi e dalla natura del set di dati oculari in considerazione.

Capitolo 4

Tecnologie utilizzate

Nel corso dello sviluppo del progetto, una delle componenti chiave è stata l'implementazione delle tecnologie giuste per garantire efficienza ed efficacia. Tra le molteplici opzioni disponibili, la scelta di adottare come linguaggio di programmazione Python si è rivelata fondamentale. La ricchezza delle librerie offerte da questo linguaggio è stato un elemento chiave nella decisione di utilizzarlo. Librerie come NumPy, Pandas, e scikit-learn forniscono strumenti potenti per la manipolazione dei dati, l'analisi statistica e l'apprendimento automatico. Questo non solo accelera lo sviluppo, ma garantisce anche la solidità e l'affidabilità delle funzionalità implementate.

4.1 Python

Python è un linguaggio di programmazione ad alto livello sviluppato dal programmatore olandese Guido van Rossum e rilasciato per la prima volta nel 1991. Caratterizzato da una sintassi semplice e intuitiva, Python supporta diversi paradigmi di programmazione: imperativo, funzionale ed object-oriented (programmazione ad oggetti).

Una volta scritto il file sorgente (file.py), questo può essere eseguito su una qualsiasi piattaforma (Mac, Windows, etc.) purché sia dotata di un interprete che si occupa di analizzare correttamente la sintassi del file e di eseguirlo, a differenza di ciò che avviene in altri linguaggi che non sono pseudocompilati, come il linguaggio C.

I suoi innumerevoli vantaggi lo rendono un linguaggio adatto a molteplici contesti:

- sviluppo Web;
- applicazioni desktop;
- sviluppo software;
- calcolo scientifico;
- sviluppo videogame;
- analisi dei dati [29];

4.2 Pandas

Pandas è una libreria open source del linguaggio Python che fornisce funzioni utili per analizzare, pulire e manipolare set di dati. Le strutture dati principalmente supportate da questa libreria sono le Series, array unidimensionali, e i Dataframe, array bidimensionali, dove i dati sono acceduti attraverso degli indici. Questi dati possono essere di vario tipo (int, float, string, etc.) e possono essere originati da molteplici fonti, come file Excel, CSV e database [30].

4.3 Seaborn

Seaborn è una libreria per la visualizzazione dati basata sulla più nota libreria Python Matplotlib. Fornisce un ampio range di funzioni per la creazione di grafici come, ad esempio, heatmaps, grafici a barre, grafici di regressione e grafici di distribuzione [**Seaborn**].

4.4 NumPy

NumPy, libreria essenziale per la computazione scientifica in Python, offre un supporto fondamentale per la gestione di array multidimensionali e funzioni matematiche avanzate. Una caratteristica chiave, particolarmente rilevante nel mio lavoro di tesi, è la capacità di eseguire operazioni di slicing. Questa funzionalità si è dimostrata preziosa, consentendo l'efficiente estrazione di porzioni specifiche di array e semplificando significativamente la manipolazione dei dati.

Capitolo 5

Metodologia e guida al Software

Questo capitolo svolge un ruolo cruciale nella comprensione dettagliata dello sviluppo del software, ponendo particolare enfasi sulle funzioni principali ritenute cruciali nel contesto del progetto di tesi. Obiettivo primario è fornire un'analisi approfondita di ciascuna di queste funzioni, delineandone lo scopo, i parametri connessi e le possibilità di personalizzazione. Ogni funzione è oggetto di un esame scrupoloso, con l'intento di rendere il capitolo una risorsa utile per chiunque dovrà interagire con il codice in futuro.

Il presente capitolo non si limiterà a fornire una semplice descrizione delle operazioni svolte, ma si addentrerà nei dettagli delle logiche implementative e delle scelte progettuali. Oltre a evidenziare le potenzialità delle funzioni, metteremo in luce anche i limiti attuali. Questa trasparenza è fondamentale per un utilizzo consapevole del software, permettendo agli utenti di comprendere i contesti in cui le funzioni possono risultare più o meno efficaci. Nella stessa ottica, affronteremo le possibili modifiche o sviluppi futuri, fornendo una roadmap che possa orientare gli utenti verso miglioramenti o personalizzazioni specifiche. Questo approccio mira a conferire agli utenti futuri una comprensione più profonda del funzionamento delle funzioni, consentendo loro di utilizzare con successo il software in diverse situazioni.

Infine, è importante sottolineare che questo capitolo non sarà solo una documentazione tecnica, ma si propone di diventare una guida pratica per coloro che dovranno utilizzare il codice in futuro. Per questo motivo verrà messo a disposizione un link al repository GitHub con tutto il materiale necessario alla comprensione del software: https://github.com/luigimiranda/Tesi-Triennale. Questo repository conterrà l'intero codice sorgente, permettendo agli utenti di esplorare ogni singola funzione per intero. Sarà quindi possibile analizzare le implementazioni e consultare commenti dettagliati utili alla comprensione del codice.

5.1 main_more_user.py

In questo file è implementata la funzione main che permette di operare su più folder contenenti più utenti e quindi di automatizzare il processo di calcolo delle fissazioni per tutti gli user.

```
1 def main():
2     extract_zip_to_folder(Path.cwd())
3     rec_dir = os.path.join(Path.cwd(), "Recordings")
```

```
    out_dir_statistiche = os.path.join(Path.cwd(), "risultati")
    os.makedirs(out_dir_statistiche, exist_ok=True)
```

Listing 5.1: Funzione main: estrazione Folder e creazione directory risultati

- **extract_zip_to_folder:** Attraversa tutti i file nella root directory. Per ogni file con estensione ".zip", verifica se è già stato estratto nella directory *Recordings*. Se non è stato estratto, la funzione apre il file ZIP e ne estrae il contenuto nella directory *Recordings*. La funzione evita di estrarre nuovamente i file già estratti, stampando un messaggio informativo.
- **rec_dir = os.path.join(Path.cwd(), "Recordings"):** Assegna il percorso completo per la directory chiamata *Recordings* alla variabile *rec_dir*.
- **out_dir_statistiche = os.path.join(Path.cwd(), "risultati"):** Simile alla riga precedente, assegna il percorso completo per una directory chiamata *risultati* all'interno della root directory in cui saranno salvati gli output di ogni user.
- **os.makedirs(out_dir_statistiche, exist_ok=True):** Questa riga crea la directory *risultati* (se non esiste già) utilizzando la funzione os.makedirs(). Il parametro exist_ok=True impedisce che venga generato un errore nel caso in cui la directory esista già.

```
for cartella_mese in os.listdir(rec_dir):
        out_dir_mese = os.path.join(out_dir_statistiche, cartella_mese)
        os.makedirs(out_dir_mese, exist_ok=True)
        all_df_statistiche = []
        users = create_user_path_dict(cartella_mese)
```

Listing 5.2: Funzione main: Ciclo for per scorrere tra i mesi

- Il *for* scorre le cartelle dei mesi presenti nella directory *Recordings* (`rec_dir`).
  - **out_dir_mese = os.path.join(out_dir_statistiche, cartella_mese):** Per ciascun mese, crea una directory corrispondente nella directory degli output *risultati* (`out_dir_statistiche`).
  - **all_df_statistiche = []:** Inizializza una lista vuota `all_df_statistiche` che sarà utilizzata per raccogliere i DataFrame statistici di tutti gli utenti nel mese corrente.
  - **users = create_user_path_dict(cartella_mese):** La funzione `create_user_path_dict` crea un dizionario con chiave l'utente e come valore un path contentente i dati utili per l'analisi; assegna il dizionario alla variabile *users*. Questo è importante perchè ci permette di mappare per ogni user i diversi folder ad esso associati e quindi scorrere in ogni path per leggere i file che serviranno per le successive funzioni.

```
for user, percorsi in users.items():
            user_directory = os.path.join(out_dir_mese, f"{user}.output")
            if not os.path.exists(user_directory):
                gazedata_presente = False
                tempi_presenti = False
                fissazioni = []
                fix_idt = []
```

Listing 5.3: Funzione main: CIclo for per scorrere tra gli user

- Il *for* attraversa ogni elemento nel dizionario *users*. Ogni elemento consiste di una chiave (user) e un valore (percorsi), dove *percorsi* è la lista di percorsi associata all'user.
- **user_directory = os.path.join(out_dir_mese, f"user.output"):** In questa riga, viene creato un percorso completo per una directory utente. Il percorso viene costruito unendo la variabile *out_dir_mes*, con il nome utente (user) e con l'estensione *.output*, infine questo percorso viene assegnato alla variabile *user_directory*.
- **if not os.path.exists(user_directory):** Questa condizione verifica se la directory utente specificata da *user_directory* non esiste. Se la directory non esiste, il blocco di codice sottostante verrà eseguito. Questa condizione è importante perché indica che se la directory utente già esiste, allora l'output associato a quell'utente è già stato elaborato in precedenza. Questa condizione è importante per evitare che venga rieseguita inutilmente l'analisi dei dati. All'interno della condizione *if*:
  - Vengono inizializzate due variabili booleane *gazedata_presente* e *tempi_presenti* a False. Questo è importante per la gestione di eventuali errori nel caso in cui un file non sia presente nella directory dell'user e quindi non potranno essere svolte alcune operazioni.
  - Vengono inizializzate le liste necessarie per memorizzare le fissazioni che verranno rilevate e analizzate.

```
for percorso in percorsi:
                    percorso_gazedata = os.path.join(percorso, "gazedata.gz")
                    percorso_tempi = os.path.join(percorso, "Tempi.txt")

                    if os.path.exists(percorso_gazedata):
                        gazedata_presente = True
                        # Esegui operazioni relative a gazedata qui
                        ...
                        # Letto il file gazedata
                        fix_idt = idt(gazedata['positionX'], gazedata['
    positionY'], gazedata['Timestamp'],dispersion, dur)
                        fix = ivt(gazedata['positionX'], gazedata['positionY'
    ], gazedata['Timestamp'],v)
                        fissazioni = fixation(gazedata['positionX'], gazedata
    ['positionY'], gazedata['Timestamp'], maxDist, minDur)

```

```
                    if os.path.exists(percorso_tempi):
                        tempi_presenti = True
                        file_path_tempi = os.path.join(percorso, "Tempi.txt")
                        tempi = read_file_txt(file_path_tempi)
```

Listing 5.4: Funzione main: Ciclo for per scorrere tra i percorsi di ogni user

- Il ciclo *for* itera attraverso i percorsi associati a ciascun utente (user) all'interno della cartella del mese (cartella_mese).
  - **percorso_gazedata = os.path.join(percorso, "gazedata.gz"):** Costruisce il percorso completo per il file "gazedata.gz" all'interno del percorso corrente.
  - **percorso_tempi = os.path.join(percorso, "Tempi.txt"):** Costruisce il percorso completo per il file "Tempi.txt" all'interno del percorso corrente.
  - **if os.path.exists(percorso_gazedata):** Verifica se il file "gazedata.gz" esiste nel percorso corrente. Se esiste:
    - Viene impostata la variabile *gazedata_presente* a True e possono essere eseguite le funzioni che utilzzano il file gazedata.gz.
    - fix_idt, fix e fissazioni sono le liste che contengono le fissazioni calcolate attraverso le relative funzioni.
  - **if os.path.exists(percorso_tempi):** Verifica se il file "Tempi.txt" esiste nel percorso corrente. Se esiste:
    - Viene impostata la variabile tempi_presenti a True
    - Legge il file Tempi.txt e salva il contenuto nella lista tempi.

```
if len(fissazioni) <= 0:
                    print(f"'{user}' non ha fissazioni")
                if gazedata_presente and tempi_presenti and len(fissazioni) > 0:
                    # Esegui operazioni che richiedono sia gazedata che Tempi
                    fix_img_map = img_detection_map(fissazioni, tempi, False)
                    prem, postm = fase_detection_dict(fix_img_map)
                    out_dir_fix = user_directory
                    filepath_csv = make_path(out_dir_fix,"info.csv")
                    df = create_df_img_fix(fix_img_map, prem, postm)
                    df.to_csv(filepath_csv)
                    df_statistiche = get_dataframe_statistics(fix_img_map, prem,postm)
                    filepath_statistiche_csv = make_path(out_dir_fix,"statistiche.csv")
                    all_df_statistiche.append(df_statistiche)

                elif not gazedata_presente or not tempi_presenti:
                    print(f"Errore: Nessun file 'gazedata' o 'Tempi' trovato per '{user}'")
```

Listing 5.5: Funzione main: Elaborazione e analisi delle fissazioni

5.1.1 Esempio di adattamento della funzione main

Questa funzione potrà essere utilizzata per studi futuri su altri dati in quanto permette di lavorare su un insieme di user senza preoccuparsi della logica dietro la ricerca, l'apertura dei folder e la lettura dei dati al loro interno. Ad esempio se si volesse lavorare sulle pupille, si potrebbe adattare facilmente come segue:

```
def main():
    ...

    for cartella_mese in os.listdir(rec_dir):
        ...

        users = create_user_path_dict(cartella_mese)
        for user, percorsi in users.items():
            user_directory = os.path.join(out_dir_mese, f"{user}.output")
            if not os.path.exists(user_directory):
                gazedata_presente = False
                tempi_presenti = False
                pupille_presente = False
                ...
                for percorso in percorsi:
                    percorso_gazedata = os.path.join(percorso, "gazedata.gz")
                    percorso_tempi = os.path.join(percorso, "Tempi.txt")
                    percorso_pupille = os.path.join(percorso, "Pupil.csv")
                    if os.path.exists(percorso_gazedata):
                        gazedata_presente = True
                        # Esegui operazioni relative a gazedata qui
                        print(f"Operazioni per '{user}' con il file gazedata:
    {percorso_gazedata}")
                        ...
                    if os.path.exists(percorso_tempi):
                        tempi_presenti = True
                        # Esegui operazioni relative a Tempi qui
                        print(f"Operazioni per '{user}' con il file Tempi: {
    percorso_tempi}")
                        ...
                    if os.path.exists(percorso_pupille):
                        pupille_presente = True
                        # Esegui le operazioni relative alle pupille
                if len(fissazioni) <= 0:
                    print(f"'{user}' non ha fissazioni")
                if gazedata_presente and tempi_presenti and len(fissazioni) >
    0:
                    # Esegui operazioni che richiedono sia gazedata che Tempi
                    print(f"Operazioni per '{user}' che richiedono sia
    gazedata che Tempi")
                elif pupille_presente and gazedata_presente:
                    # Esegui operazioni che richiedono pupille e gazedata
                elif not gazedata_presente or not tempi_presenti:
```

```
40                      print(f"Errore: Nessun file 'gazedata' o 'Tempi' trovato
    per '{user}'")
41
42          ...
43
44          ...
```

Listing 5.6: Funzione main modificata

Aggiungendo quindi le linee di codice evidenziate in giallo si può lavorare facilmente con i file relativi alle pupille per ogni user. In generale basta impostare una variabile booleana a False relativa al file su cui vogliamo lavorare prima di ciclare nei percorsi di ogni user, appena si entra in un percorso in cui è presente il file desiderato, si imposta questa variabile booleana a True e si eseguono le operazioni sul file desiderato; in questo modo siamo sicuri che in una qualunque sotto cartella dello user i-esimo abbiamo trovato o meno il file desiderato. Se abbiamo bisogno di operazioni che includono l'utilizzo di più file basta aggiungere la condizione che controlla se le variabili booleane relative a quei file sono a True: in caso affermativo allora potranno essere eseguite operazioni che richiedono l'uso obbligatorio e contemporaneo dei dati ricavati da due file distinti. Ad esempio, per le fissazioni c'è bisogno sia del file gazedata sia del file tempi.

5.2 detection_fase_cognitivata.py

```
1  def img_detection_map(fixList, tempiList, restrict_choice):
2
3      riallinea_tempi(tempiList)
4      img_that_gen_fix = OrderedDict()  # Usiamo un dizionario ordinato
5
6      for time in tempiList:
7          img_that_gen_fix[tuple(time)] = []  # Inizializziamo il dizionario
       con chiavi vuote
8
9      for fix in fixList:
10         for i, time in enumerate(tempiList):
11             if restrict_choice:
12                 if fix[0] < time[2] and fix[0] >= time[1]:
13                     img_that_gen_fix[tuple(time)].append(fix)
14                     break
15             else:
16                 if fix[0] < time[2] and fix[0] < time[1]:
17                     if i > 0:
18                         prev_time = tempiList[i-1]
19                         img_that_gen_fix[tuple(prev_time)].append(fix)
20                     break
21                 if fix[0] < time[2] and fix[0] >= time[1]:
22                     img_that_gen_fix[tuple(time)].append(fix)
23                     break
24
25     remove_not_util_map(img_that_gen_fix)
```

```
    return img_that_gen_fix
```

Listing 5.7: Funzione image_detection_map

Questa funzione crea un dizionario in cui la chiave è lo stimolo, mentre il valore è una lista di fissazioni ad esso associate. In particolare questa funzione prende in input la lista di fissazioni, la lista degli stimoli e permette di associare ad ogni stimolo quali sono le fissazioni che ha generato. Durante la fase di associazione di stimoli e fissazioni è stato notato che alcune fissazioni iniziano con uno stimolo e finiscono allo stimolo successivo; quindi è stato aggiunto il parametro booleano restrict_choice che permette due calcoli diversi:

- Se è True considera solo le fissazioni che sono comprese in un intervallo di uno degli stimoli.
- Se è False considera le fissazioni che iniziano con uno stimolo e finiscono allo stimolo successivo come fissazioni generate dallo stimolo precedente.

Infatti, come possiamo notare nel codice, nel caso in cui restrict_choice è True le fissazioni che non sono comprese in nessuno intervallo di stimoli vengono elimante, invece nel caso in cui è False non si perde nessuna fissazione.

Un'altra importante funzione è quella che ci permette di distinguere le fissazioni generate dagli stimoli nella sola fase pre-cognitiva e post-cognitiva. Questa funzione prende in input il dizionario generato dalla funzione 5.7, inizializza due dizionari per le due fasi che hanno sempre come chiave lo stimolo e come valore una lista di fissazioni; scorre tutte le chiavi e per ogni chiave scorre ogni fissazione: le fissazioni al di sotto dei 0.3 secondi sono aggiunte al dizionario per la fase pre-cognitiva, quelle al di sopra sono aggiunte al dizionario per la fase post-cognitiva.

```
def fase_detection_dict(img_that_gen_fix):
    pre_cognitiva = {}
    post_cognitiva = {}

    for chiave, fissazioni in img_that_gen_fix.items():
        pre_fissazioni = []
        post_fissazioni = []
        secondo_valore_chiave = chiave[1]

        for fissazione in fissazioni:
            primo_valore_fissazione = fissazione[0]
            differenza = primo_valore_fissazione - secondo_valore_chiave

            if differenza <= 0.3:
                pre_fissazioni.append(fissazione)
            else:
                post_fissazioni.append(fissazione)

        if pre_fissazioni:
            pre_cognitiva[chiave] = pre_fissazioni
        if post_fissazioni:
```

```
            post_cognitiva[chiave] = post_fissazioni

    return pre_cognitiva, post_cognitiva
```

Listing 5.8: Funzione fase_detection_dict

Queste due funzione sono molto utili perché permetteranno poi di generare i dataframe e quindi i file csv relativi alle statistiche per ogni user.

5.3 fixation_detection.py

In questo file sono presenti le implementazioni degli algoritmi I-VT e I-DT per la rilevazione delle fissazioni.

L' algoritmo I-VT è stato implementato partendo dallo pseudocodice esposto da Salvucci e Goldberg [21]. La funzione prende in input:

- **x**: lista delle coordinate x del parametro gaze2d del file gazedata.
- **y**: lista delle coordinate y del parametro gaze2d del file gazedata.
- **time**: lista dei tempi di acquisizione relativi al parametro timestamp del file gazedata.
- **velocity_threshold**: soglia della velocità per distinguere le fissazioni dalle saccadi.

La funzione restituisce in output la lista di fissazioni, indicando tempo di inizio, tempo di fine, durata, centroide x, centroide y. Inizialmente vengono calcolate le velocità punto-punto(coordinate) con la funzione di appoggio calculate_velocity. Etichetta ogni punto come fissazioni o saccadi in base a una soglia di velocità, raggruppa i punti di fissazione consecutivi in gruppi di fissazione, e infine restituisce una lista di tuple contenenti informazioni su ciascuna fissazione individuata, includendo inizio, fine, durata e coordinate del centroide. La durata della fissazione deve essere superiore a 0.2 secondi per essere considerata.

```
def ivt(x, y, time, velocity_threshold):
    # Calculate velocities
    velocities = calculate_velocity(x, y, time)

    # Label points as fixation or saccade
    labels = np.where(velocities <= velocity_threshold, 'Fixation', 'Saccade'
    )

    # Collapse consecutive fixation points into fixation groups
    fixation_groups = []
    current_group = []
    for i, label in enumerate(labels):
        if label == 'Fixation':
            current_group.append(i)
        else:
            if current_group:
                fixation_groups.append(current_group)
                current_group = []
```

```
18
19      # Map each fixation group to a fixation at the centroid
20      fixations = []
21      for group in fixation_groups:
22          group_x = x[group]
23          group_y = y[group]
24          start_time = time[group[0]]
25          end_time = time[group[-1]]
26          duration = end_time - start_time
27          if duration >= 0.2:
28              centroid_x = np.mean(group_x)
29              centroid_y = np.mean(group_y)
30              fixations.append((start_time, end_time, duration, centroid_x,
    centroid_y))
31
32      return fixations
```

Listing 5.9: Algoritmo I-VT

Per l'algoritmo I-DT, invece, abbiamo due implementazioni simili ma che differiscono per la complessità computazionale. Entrambe le funzioni prendono in input:

- **x**: lista delle coordinate x del parametro gaze2d del file gazedata.
- **y**: lista delle coordinate y del parametro gaze2d del file gazedata.
- **time**: lista dei tempi di acquisizione relativi al parametro timestamp del file gazedata.
- **dispersion_threshold**: soglia di dispersione per distinguere le fissazioni dalle saccadi.
- **duration_threshold**: soglia della durata, utile per definire la grandezza della finestra scorrevole per il campionamento delle fissazioni.

Le funzioni restituiscono in output la lista di fissazioni, indicando per ciascuna fissazione tempo di inizio, tempo di fine, durata, centroide x, centroide y.

```
1  def idt(x, y, time, dispersion_threshold, duration_threshold):
2      fixations = []
3      points = np.array(list(zip(x, y, time)))
4
5      while points.shape[0] > 0:
6          window = points[:duration_threshold, :]
7          x_values = window[:, 0]
8          y_values = window[:, 1]
9
10         dispersion = (np.max(x_values) - np.min(x_values)) + (np.max(y_values
    ) - np.min(y_values))
11
12         if dispersion <= dispersion_threshold:
13             while dispersion <= dispersion_threshold and window.shape[0] <
    points.shape[0]:
14                 window = np.vstack((window, points[window.shape[0], :]))
15                 x_values = window[:, 0]
```

```
                y_values = window[:, 1]
                dispersion = (np.max(x_values) - np.min(x_values)) + (np.max(
    y_values) - np.min(y_values))

            if window.shape[0] >= duration_threshold:  # Assicurati che ci
    siano almeno due punti in finestra
                centroid_time = np.mean(window[:, 2])
                start_time = window[0, 2]
                end_time = window[-1, 2]
                duration = end_time - start_time
                pos_x = np.mean(window[:, 0])
                pos_y = np.mean(window[:, 1])

                fixations.append((start_time, end_time, duration, pos_x,
    pos_y))

            points = points[window.shape[0]:, :]
        else:
            points = points[1:, :]

    return fixations
```

Listing 5.10: Algoritmo I-DT veloce

```
def idtlento(x, y, t, dispersion, duration):
    fixations = []

    start = 0  # Posizione iniziale della finestra

    while start <= len(x) - duration:
        end = start + duration  # Posizione finale della finestra

        # Creazione della finestra
        x_win = x[start:end]
        y_win = y[start:end]

        # Calcolo della dispersione
        D = (np.max(x_win, axis=0) - np.min(x_win, axis=0)) + \
            (np.max(y_win, axis=0) - np.min(y_win, axis=0))

        j = 1  # Espansore della finestra

        while D <= dispersion and end + j < len(x):
            # Espansione della finestra
            x_win = x[start:(end + j)]
            y_win = y[start:(end + j)]

            D = (np.max(x_win, axis=0) - np.min(x_win, axis=0)) + \
                (np.max(y_win, axis=0) - np.min(y_win, axis=0))

            if D > dispersion:
                # Selezione della finestra (j - 1) come fissazione
```

```
                fixations.append([t[start], t[end + j - 1], t[end + j - 1] - t[start], np.mean(x_win[:-1]), np.mean(y_win[:-1])])
                start = end + j  # Salta i punti della finestra
                break
            elif end + j == len(x):
                # Gestisci l'ultima finestra se i dati terminano durante una fissazione
                fixations.append([t[start], t[end], t[end] - t[start], np.mean(x_win), np.mean(y_win)])
                start = end + j
                break

            j += 1

        start += 1

    return fixations
```

Listing 5.11: Algoritmo I-DT lento

Entrambe le funzioni lavorano con una finestra scorrevole che si muove tra i campioni e definisce se in una finestra c'è o meno una fissazione. In particolare il flusso dell'algoritmo è il seguente:

- Crea a partire dal primo campione una finestra di dimensione duration_threshld.
- Calcola la dispersione delle coordinate x e y all'interno di questa finestra.
- Se la dispersione calcolata è <= della dispersion_threshold allora quella finestra rappresenta una fissazione.
- La finestra viene incrementata di uno verso destra finchè la dispersione calcolata è <= della dispersion_threshold. Una volta superata la soglia di dispersione la funzione calcola i centroidi di x e y, il tempo di inizio, di fine e la durata della fissazione. Infine aggiunge la fissazione alla lista fixations.
- Se la dispersione calcolata è > della dispersion_threshold allora quella finestra non rappresenta una fissazione. La finestra si sposta al campione successivo e riparte con la creazione della finestra.

La differenza sostanziale è evidenziata in giallo nel codice. In particolare nell'I-DT lento la finestra avanza sempre di un punto alla volta con l'istruzione $start+ = 1$; invece nell'I-DT veloce la finestra si sposta ogni volta di un numero di campioni pari alla duration_threshold. In sostanza, I-DT lento crea una finestra di dimensione n a partire dal primo campione, amplia la finestra se necessario e dopo aver definito una fissazione si sposta al secondo campione e ricrea la finestra di dimensione n. Invece, I-DT veloce crea una finestra di dimensione n a partire dal primo campione, se necessario amplia la finestra e dopo aver definito una fissazione si sposta della dimensione n, ovvero al campione n.

```
    points = points[window.shape[0]:, :]
```

La linea di codice ha lo scopo di aggiornare l'array points escludendo le righe che sono già state considerate all'interno della finestra corrente (window). Questa operazione permette di avanzare direttamente al successivo punto non esaminato.

Questa operazione è cruciale per garantire che ogni punto venga considerato solo una volta nell'analisi e che l'algoritmo possa continuare a valutare i punti rimanenti nel set di dati.

Capitolo 6

Risultati

In questo capitolo sono esposti i risultati ottenuti mediante l'implementazione e l'esecuzione degli algoritmi nell'ambito della nostra ricerca. Per garantire una valutazione equa, tutti gli algoritmi sono stati eseguiti su un set di dati uniforme. In particolare sono stati utilizzati dei parametri sperimentali per:

- Soglia di velocità nell'algoritmo I-VT.
- Soglia di dispersione nell'algoritmo I-DT.
- Soglia di durata in tutti gli algoritmi.

Per la soglia di durata sono stati utilizzati tre valori differenti per identificare la durata minima di una fissazione:

- 60 ms: valore estrapolato dalla documentazione del Tobii.
- !00 ms e 200 ms: valori medi di una fissazione [21].

I risultati degli algoritmi contengono molti parametri, tuttavia in questo capitolo ci concentreremo solo su alcuni di essi.

In generale ogni algoritmo è stato eseguito tre volte e si è notato che modificando il parametro relativo alla durata minima di una fissazione i risultati sono considerevolmente differenti. Prenderemo come esempio l'algoritmo I-DT per evidenziare queste differenze nei risultati.

User-ID	TotFix	TotFixStimoli	#medioFissazioni	#medioFissazioniStimoli
53	148	65	1.7011494252873562	1.625
54	114	28	1.3103448275862069	0.7
55	5	0	0.0416666666666666	0.0
56	2	0	0.0227272727272727	0.0
57	99	37	0.8048780487804879	0.6607142857142857
58	33	10	0.2727272727272727	0.1785714285714285
61	3	1	0.0252100840336134	0.0178571428571428
62	10	0	0.08130081300813	0.0
29	497	326	4.073770491803279	5.821428571428571
30	637	378	5.308333333333334	6.75
31	325	203	2.6422764227642275	3.625
32	57	21	0.456	0.375
33	742	459	6.132231404958677	8.196428571428571
34	296	168	2.446280991735537	3.0
35	88	51	0.7154471544715447	0.9107142857142856
36	196	132	1.6333333333333333	2.357142857142857
37	435	254	5.0	6.35
38	381	268	4.233333333333333	6.7
39	103	66	1.1839080459770115	1.65
40	99	67	0.8048780487804879	1.1964285714285714
41	22	12	0.1833333333333333	0.2142857142857142
22	123	64	1.0081967213114753	1.1428571428571428
23	571	361	6.6395348837209305	9.025
24	59	42	0.4645669291338583	0.75
25	434	291	3.557377049180328	5.196428571428571
26	418	276	3.4833333333333334	4.928571428571429
27	265	154	2.190082644628099	2.75
28	788	504	6.40650406504065	9.0
28	788	504	6.40650406504065	9.0
60	5	0	0.042016806722689	0.0
63	49	27	0.392	0.4821428571428571
66	5	0	0.042016806722689	0.0
42	363	228	2.951219512195122	4.071428571428571
44	734	468	5.967479674796748	8.357142857142858
45	32	0	0.3720930232558139	0.0
46	269	179	2.2416666666666667	3.196428571428572
47	270	147	2.1951219512195124	2.625
48	1	0	0.0081967213114754	0.0
50	47	17	0.3916666666666666	0.3035714285714285
51	128	50	1.0578512396694215	0.8928571428571429
52	18	2	0.15	0.0357142857142857

Tabella 6.1: Risultati I-DT con dispersione 0.01 e durata minima 200 ms.

User-ID	TotFix	TotFixStimoli	#medioFissazioni	#medioFissazioniStimoli
53	436	218	5.011494252873563	5.45
54	400	181	4.597701149425287	4.525
55	39	11	0.325	0.1964285714285714
56	29	5	0.3295454545454545	0.125
57	295	145	2.398373983739837	2.5892857142857144
58	119	48	0.9834710743801652	0.8571428571428571
59	17	0	0.1954022988505747	0.0
60	4	0	0.032258064516129	0.0
61	24	6	0.2016806722689075	0.1071428571428571
62	92	12	0.7479674796747967	0.2142857142857142
29	1222	744	10.01639344262295	13.285714285714286
30	994	609	8.283333333333333	10.875
31	693	422	5.634146341463414	7.535714285714286
32	219	110	1.752	1.9642857142857144
33	1350	812	11.15702479338843	14.5
34	834	493	6.892561983471074	8.803571428571429
35	444	245	3.609756097560976	4.375
36	679	435	5.658333333333333	7.767857142857143
37	994	595	11.42528735632184	14.875
38	889	565	9.877777777777778	14.125
39	448	255	5.149425287356322	6.375
40	447	264	3.634146341463415	4.714285714285714
41	151	85	1.2583333333333333	1.5178571428571428
22	500	285	4.098360655737705	5.089285714285714
23	848	517	9.86046511627907	12.925
24	467	287	3.677165354330709	5.125
25	1174	751	9.62295081967213	13.410714285714286
26	869	556	7.241666666666666	9.928571428571429
27	669	392	5.528925619834711	7.0
28	1130	705	9.1869918699187	12.589285714285714
28	1130	705	9.1869918699187	12.589285714285714
60	42	9	0.3529411764705882	0.1607142857142857
63	190	92	1.52	1.6428571428571428
66	42	9	0.3529411764705882	0.1607142857142857
42	894	522	7.268292682926829	9.321428571428571
44	1220	804	9.91869918699187	14.357142857142858
45	62	0	0.7209302325581395	0.0
46	705	452	5.875	8.071428571428571
47	763	449	6.203252032520325	8.017857142857142
48	3	1	0.0245901639344262	0.0178571428571428
49	3	1	0.024390243902439	0.0178571428571428
50	119	30	0.9916666666666668	0.5357142857142857
51	366	170	3.024793388429752	3.0357142857142856
52	91	25	0.7583333333333333	0.4464285714285714

Tabella 6.2: Risultati I-DT con dispersione 0.01 e durata minima 100 ms.

User-ID	TotFix	TotFixStimoli	#medioFissazioni	#medioFissazioniStimoli
53	824	464	9.471264367816092	11.6
54	874	479	10.045977011494251	11.975
55	143	50	1.1916666666666669	0.8928571428571429
56	132	31	1.5	0.775
57	705	380	5.7317073170731705	6.785714285714286
58	290	132	2.396694214876033	2.357142857142857
59	67	8	0.7701149425287356	0.2
60	23	0	0.1854838709677419	0.0
61	122	45	1.0252100840336134	0.8035714285714286
62	285	95	2.317073170731707	1.6964285714285714
29	1741	1077	14.270491803278688	19.232142857142858
30	1232	757	10.266666666666667	13.517857142857142
31	1345	834	10.934959349593496	14.892857142857142
32	664	363	5.312	6.482142857142857
33	1565	933	12.93388429752066	16.660714285714285
34	1453	885	12.008264462809915	15.803571428571429
35	999	563	8.121951219512194	10.053571428571429
36	1236	769	10.3	13.732142857142858
37	1432	874	16.45977011494253	21.85
38	1322	809	14.688888888888888	20.225
39	982	555	11.28735632183908	13.875
40	1022	596	8.308943089430894	10.642857142857142
41	500	290	4.166666666666667	5.178571428571429
22	1087	628	8.90983606557377	11.214285714285714
23	978	569	11.372093023255816	14.225
24	1287	794	10.133858267716535	14.178571428571429
25	1864	1172	15.278688524590164	20.928571428571427
26	1336	844	11.133333333333333	15.071428571428571
27	1136	673	9.388429752066116	12.017857142857142
28	1312	798	10.666666666666666	14.25
28	1312	798	10.666666666666666	14.25
60	166	37	1.3949579831932772	0.6607142857142857
63	382	187	3.056	3.3392857142857144
66	166	37	1.3949579831932772	0.6607142857142857
42	1513	899	12.30081300813008	16.053571428571427
44	1520	1007	12.357723577235772	17.982142857142858
45	91	3	1.058139534883721	0.075
46	1306	829	10.883333333333333	14.803571428571429
47	1479	891	12.024390243902438	15.910714285714286
48	7	2	0.0573770491803278	0.0357142857142857
49	29	9	0.2357723577235772	0.1607142857142857
50	231	55	1.925	0.9821428571428572
51	710	380	5.867768595041323	6.785714285714286
52	218	65	1.8166666666666669	1.1607142857142858

Tabella 6.3: Risultati I-DT con dispersione 0.01 e durata minima 60 ms.

Lo stesso risultato è stato riscontrato per l'algoritmo I-VT e PyGaze. Nello specifico diminuendo il valore della durata minima di una fissazione, si sta rendendo l'algoritmo più sensibile ai movimenti oculari più brevi. Questo comporta un'identificazione e una registrazione più dettagliata delle fissazioni, con un conseguente aumento significativo del numero di fissazioni rilevate quando si passa da 200 ms a 100 ms e poi a 60 ms. Tale sensibilità crescente può rendere lo studio più adatto per la rilevazione di sguardi più brevi e dettagliati, ma è importante bilanciare la sensibilità con la specificità in base agli obiettivi dello studio.

6.1 Analisi dei task

Per il progetto Parkinson risulta molto utile l'analisi degli specifici task che vengono mostrati ai pazienti nella fase di acquisizione dei dati relativi allo sguardo. Per questo motivo verranno presentate le differenze nei risultati dei tre algoritmi. In questa analisi abbiamo preferito considera come durata minima di 60 ms per restare in linea con le specifiche delle documentazione ufficiale del Tobii.

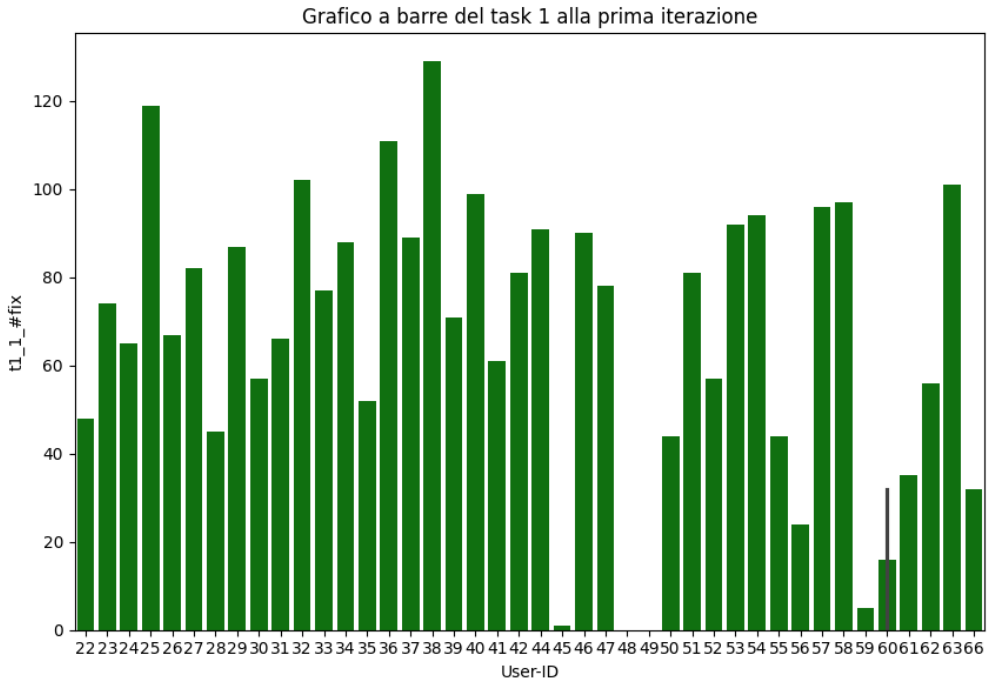

Figura 6.1: Istogramma del numero di fissazioni per il task 1 con algoritmo I-DT. Durata minima fissazione 60 ms.

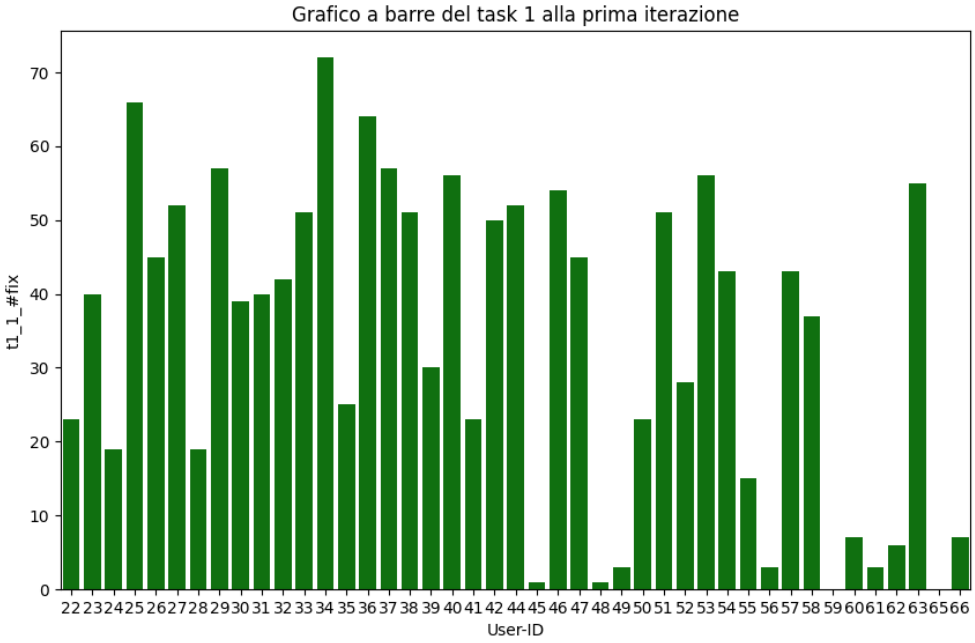

Figura 6.2: Istogramma del numero di fissazioni per il task 1 con algoritmo I-VT. Durata minima fissazione 60 ms.

Possiamo notare come l'algoritmo I-DT risulti più accurato rispetto a I-VT e riesca a rilevare un numero maggiore di fissazioni. Un altro importante confronto può essere effettuato sul Time To First Fixation di ogni task mostrato ai pazienti.

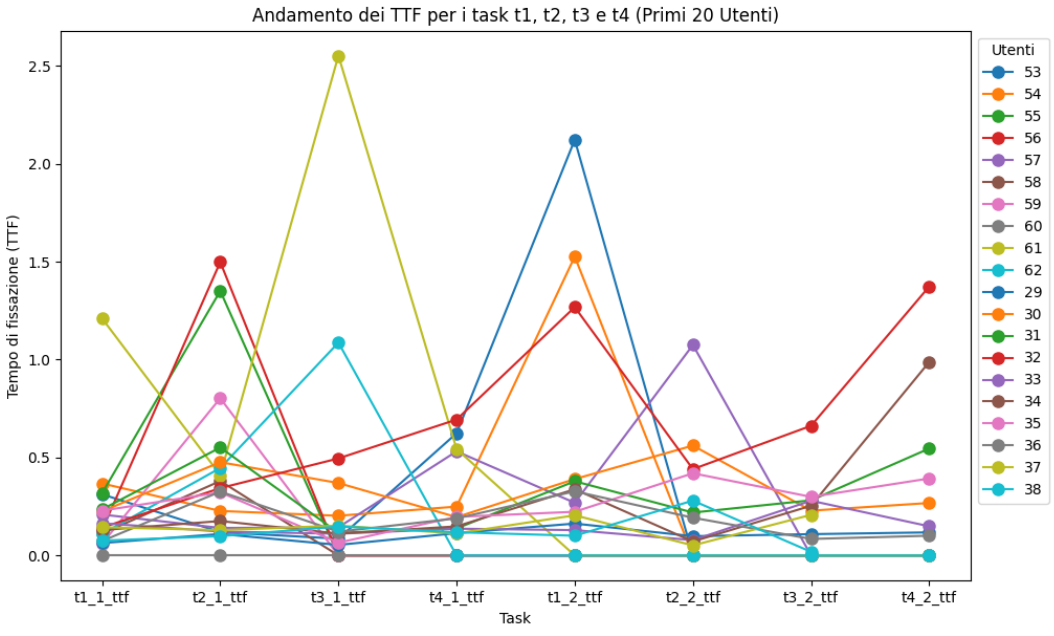

Figura 6.3: Grafico a linea per evidenziare l'andamento dei TTF in ogni task con I-DT. Durata minima fissazione 60 ms.

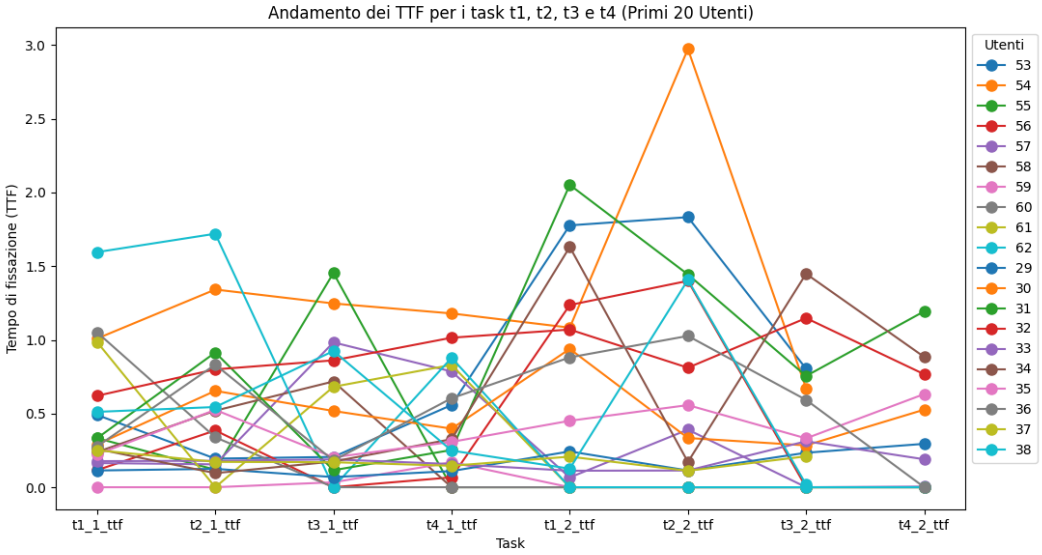

Figura 6.4: Grafico a linea per evidenziare l'andamento dei TTF in ogni task con I-DT. Durata minima fissazione 60 ms.

Nel corso di questa ricerca, si sono riscontrate notevoli disparità nei calcoli dei Time to First Fixation (TTF). Queste discrepanze sono presumibilmente imputabili ai parametri adottati per la dispersione in I-DT e la velocità in I-VT. Per garantire un'analisi più coesa e approfondita, risulta sufficiente condurre un calcolo sperimentale accurato di entrambe le soglie. Un approccio sperimentale mirato e coerente nel determinare tali parametri potrebbe contribuire significativamente a mitigare le differenze osservate nei TTF, garantendo risultati più omogenei e affidabili nell'ambito dell'analisi dell'attenzione visiva.

6.2 Risultati dettagliati per specifici pazienti

Grazie all'analisi statistica condotta in questa fase, è ora possibile confrontare le reazioni di diversi pazienti di fronte alla visualizzazione dei quattro task. Inizialmente, verranno presentate le differenze riscontrate nello stesso utente, calcolate utilizzando sia l'algoritmo I-DT che I-VT. Successivamente, verranno esaminate le disparità tra un paziente sano e uno malato.

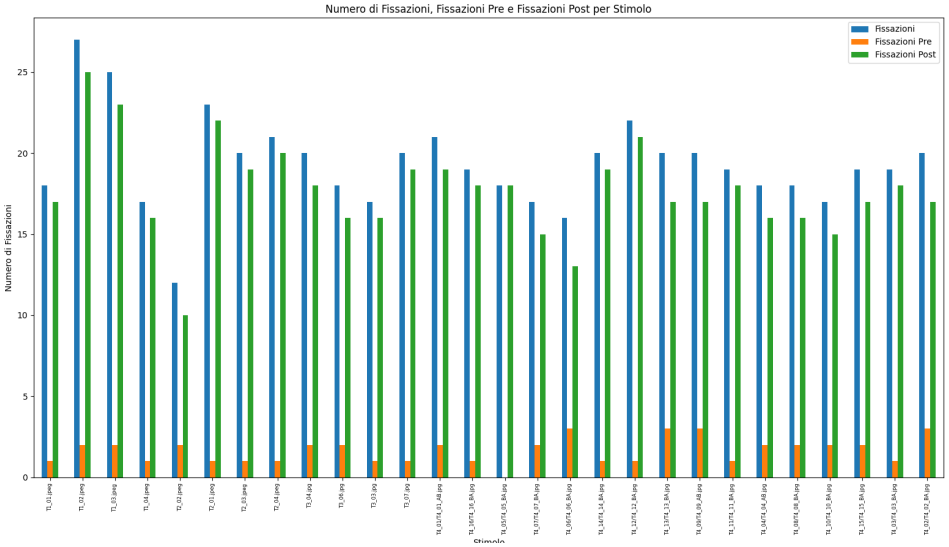

Figura 6.5: Istogramma.

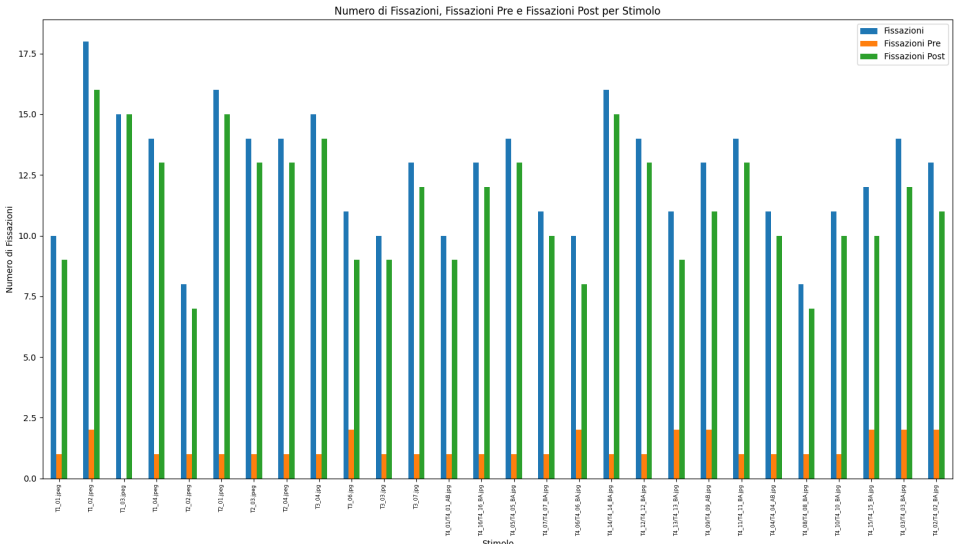

Figura 6.6: Istogramma.

I grafici saranno focalizzati sulle differenze nel numero di fissazioni cosi suddivise:

- Totale delle Fissazioni: Rappresenta l'intero periodo di osservazione.
- Fissazioni fase Pre-Cognitiva (Primi 300 ms): Mostra le fissazioni avvenute nei primi 300 millisecondi dall'inizio dell'osservazione.
- Fissazioni fase Post-Cognitiva (Dopo i 300 ms): Rappresenta le fissazioni che si sono verificate dopo il primo intervallo di 300 millisecondi.

Questo approccio segmentato permette di analizzare in dettaglio come l'attenzione visiva dei pazienti varia durante le diverse fasi dell'elaborazione cognitiva, contribuendo a una comprensione più approfondita dei processi visivi coinvolti nei quattro task proposti.

Infine mostriamo su tutte le acquisizione effettuate nel 2023 mostrano la differenza nella durata media delle fissazioni con I-DT e I-VT.

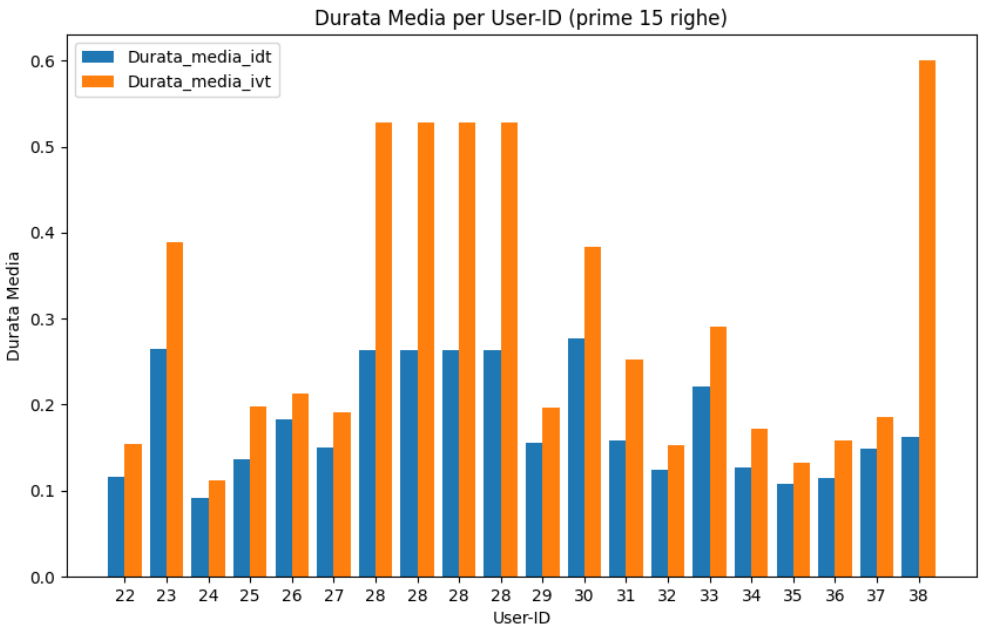

Figura 6.7: Istogramma per durata media fissazioni con I-DT e I-VT.

Come esposto nei capitoli precedenti, la durata media di una fissazione si colloca solitamente nell'intervallo compreso tra 100 e 300 millisecondi. Tuttavia, l'algoritmo I-VT, in diverse acquisizioni, con nel grafico 6.7, mostra dati che potrebbero risultare poco affidabili, compromettendo così la precisione complessiva del sistema di tracciamento oculare. Questa discrepanza potrebbe derivare dalla definizione del parametro di soglia di velocità, probabilmente non completamente coerente con l'utilizzo specifico dell'eye tracker Tobii Pro Glasses 3.

Inoltre, bisogna tenere presente che anche per l'algoritmo I-DT è necessario determinare sperimentalmente un valore adeguato per la soglia di dispersione. Questo passo è cruciale per garantire una calibrazione ottimale dell'eye tracker e per assicurare la coerenza e l'affidabilità dei dati acquisiti durante le registrazioni oculari.

Capitolo 7

Conclusioni

L'analisi sperimentale dei dati raccolti dall'Eyetracker Tobii Pro Glasses 3 presentata all'interno di questo elaborato rappresenta un punto di partenza per studi futuri.

Il presente lavoro di tesi costituisce un avanzamento nell'approfondimento dello studio delle fissazioni, focalizzandosi sull'analisi degli algoritmi essenziali per la rilevazione di tali fenomeni oculari e ponendo l'attenzione alla loro integrazione con l'eye tracker Tobii Pro Glasses 3. Esplorando gli algoritmi esistenti, è emersa l'importanza di definire correttamente parametri sperimentali come la velocità, la dispersione e la distanza tra i punti di fissazione acquisiti, specialmente in relazione alle caratteristiche specifiche dell'eye tracker utilizzato; il controllo e la revisione di tali parametri è fondamentale per ottenere risultati coerenti e attendibili.

Inoltre, la progettazione di un codice automatizzato non solo ha semplificato il processo di acquisizione dei dati, ma ha anche fornito una struttura solida per la loro lettura e la creazione di dataframe per l'analisi statistica. Questa automazione consentirà a coloro che in futuro approfondiranno queste tematiche di concentrarsi esclusivamente sullo studio delle fissazioni, avendo già a disposizione gli strumenti necessari per elaborare e interpretare i dati.

7.1 Sviluppi futuri

Una prospettiva di sviluppo interessante potrebbe riguardare l'integrazione dello studio delle fissazioni con altre metriche oculari, come le saccadi e le pupille. L'utilizzo combinato di tali informazioni potrebbe aprire nuove possibilità di ricerca, ad esempio nell'identificazione di pattern distintivi per individuare soggetti affetti da patologie come il morbo di Parkinson da quelli in uno stato di salute normale.

In conclusione, questo lavoro di tesi ha fornito una base solida sia per l'analisi delle fissazioni che per future esplorazioni e approfondimenti in questo campo di ricerca, aprendo nuove vie per la comprensione dei processi oculari e le loro implicazioni cliniche.

Bibliografia

[1] https://www.facebook.com/NIHAging. *Parkinson's Disease: Causes, Symptoms, and Treatments*. en. URL: https://www.nia.nih.gov/health/parkinsons-disease (visitato il 27/06/2023).

[2] Samuel Stuart et al. "Pro-saccades predict cognitive decline in Parkinson's disease: ICICLE-PD". In: *Movement Disorders* 34.11 (2019), pp. 1690–1698.

[3] J. M. Gibson, R. Pimlott e C. Kennard. "Ocular motor and manual tracking in Parkinson's disease and the effect of treatment". eng. In: *Journal of Neurology, Neurosurgery, and Psychiatry* 50.7 (lug. 1987), pp. 853–860. ISSN: 0022-3050. DOI: 10.1136/jnnp.50.7.853.

[4] *Characterizing the Relationship Between Eye Movement Parameters and Cognitive Functions in Non-demented Parkinson's Disease Patients with Eye Tracking | Protocol (Translated to Italian)*. en. URL: https://www.jove.com/v/60052/characterizing-relationship-between-eye-movement-parameters-cognitive?language=Italian (visitato il 18/08/2023).

[5] *CHE COS'È L'EYE TRACKING*. it-IT. Mag. 2019. URL: https://www.srlabs.it/che-cose-leye-tracking/ (visitato il 18/08/2023).

[6] Yasuo TERAO, Hideki FUKUDA e Okihide HIKOSAKA. "What do eye movements tell us about patients with neurological disorders? — An introduction to saccade recording in the clinical setting —". In: *Proceedings of the Japan Academy. Series B, Physical and Biological Sciences* 93.10 (dic. 2017), pp. 772–801. ISSN: 0386-2208. DOI: 10.2183/pjab.93.049. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5790757/ (visitato il 18/08/2023).

[7] Michael K Tanenhaus e Michael J Spivey-Knowlton. "Eye-tracking". In: *Language and Cognitive processes* 11.6 (1996), pp. 583–588.

[8] Vidas Raudonis, Rimvydas Simutis e Gintautas Narvydas. "Discrete eye tracking for medical applications". In: *2009 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies*. IEEE. 2009, pp. 1–6.

[9] Susan Stuart et al. "Accuracy and re-test reliability of mobile eye-tracking in Parkinson's disease and older adults". In: *Medical engineering & physics* 38.3 (2016), pp. 308–315.

[10] Benfatto MN et al. Tsitsi P. “Fixation Duration and Pupil Size as Diagnostic Tools in Parkinson’s Disease”. In: *Journal of Parkinson’s Disease* (2021). URL: https://content.iospress.com/articles/journal-of-parkinsons-disease/jpd202427.

[11] *Oculomotor Performances Are Associated With Motor and Non-motor Symptoms in Parkinson’s Disease*. URL: https://www.frontiersin.org/articles/10.3389/fneur.2018.00960/full.

[12] Panagiota Tsitsi et al. “Pupil light reflex dynamics in Parkinson’s disease”. In: *Frontiers in Integrative Neuroscience* 17 (2023). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10506153/.

[13] Roy S. Hessels et al. “Is the eye-movement field confused about fixations and saccades? A survey among 124 researchers”. In: *Royal Society Open Science* 5.8 (ago. 2018), p. 180502. ISSN: 2054-5703. DOI: 10.1098/rsos.180502. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6124022/ (visitato il 14/07/2023).

[14] Luca Mastrogiacomo e Riccardo Gervasi. “Valutazione dello stress mentale e del carico di lavoro cognitivo attraverso i dati di eye-tracking”. it. In: ().

[15] Tesi di Laurea. “ACUTEZZA VISIVA CON SGUARDO PREFERENZIALE”. it. In: ().

[16] AuthorLuca Vannetiello. *Fissazione Dello Sguardo e I Movimenti Oculari Rapidi Spontanei e Incontrollati*. it-IT. Mag. 2015. URL: https://lucavannetiello.com/2015/05/13/fissazione-dello-sguardo-e-i-movimenti-oculari-rapidi-spontanei-e-incontrollati/ (visitato il 19/08/2023).

[17] Bryn Farnsworth. *What is VR Eye Tracking? [And How Does it Work?]* URL: https://imotions.com/blog/learning/best-practice/vr-eye-tracking/.

[18] Bryn Farnsworth. *Eye Tracking: The Complete Pocket Guide*. URL: https://imotions.com/blog/learning/best-practice/eye-tracking/.

[19] *Tobii Pro Glasses 3*. URL: https://www.tobii.com/.

[20] Ramakrishna S Pillalamarri et al. “Cluster: A program for the identification of eye-fixation-cluster characteristics”. In: *Behavior Research Methods, Instruments, & Computers* 25 (1993), pp. 9–15.

[21] Dario D Salvucci e Joseph H Goldberg. “Identifying fixations and saccades in eye-tracking protocols”. In: *Proceedings of the 2000 symposium on Eye tracking research & applications*. 2000, pp. 71–78.

[22] Casper J Erkelens e Ingrid MLC Vogels. “The initial direction and landing position of saccades”. In: *Studies in Visual Information Processing*. Vol. 6. Elsevier, 1995, pp. 133–144.

[23] Tayyar Sen e Ted Megaw. “The effects of task variables and prolonged performance on saccadic eye movement parameters”. In: *Advances in Psychology*. Vol. 22. Elsevier, 1984, pp. 103–111.

[24] Dario D Salvucci. “An interactive model-based environment for eye-movement protocol analysis and visualization”. In: *Proceedings of the 2000 symposium on eye tracking research & applications*. 2000, pp. 57–63.

[25] Dario Dino Salvucci. *Mapping eye movements to cognitive processes*. Carnegie Mellon University, 1999.

[26] Lawrence R Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[27] Heino Widdel. “Operational problems in analysing eye movements”. In: *Advances in psychology*. Vol. 22. Elsevier, 1984, pp. 21–29.

[28] Rudy Den Buurman, Theo Roersema e Jack F Gerrissen. “Eye movements and the perceptual span in reading”. In: *Reading Research Quarterly* (1981), pp. 227–235.

[29] *Cos'è Python*. URL: `https://www.python.it/`.

[30] *pandas*. URL: `https://pandas.pydata.org/`.

Ringraziamenti

Desidero ringraziare il professore Andrea Francesco Abate per avermi dato l'opportunità di partecipare ad un progetto formativo e intenso e per la sua disponibilità.

Ringrazio inoltre la dottoressa Lucia Cascone per la sua preziosa guida durante la stesura della tesi, la sua competenza e il suo supporto sono stati fondamentali in questo percorso.
Questo percorso è stato più complicato di quanto mi aspettassi e quindi vorrei ringraziare tutti coloro che mi sono stati vicini anche nei momenti più intensi e difficili. Sono grato di aver avuto persone come voi al mio fianco; questo traguardo è anche merito vostro!

Ringrazio la mia famiglia per essermi stata accanto sempre, per aver creduto in me e nelle mie scelte. A Rino, un padre e un fratello. Ti ringrazio per il supporto e per i tuoi consigli, mai banali, che mi hanno permesso di superare ogni ostacolo che si è presentato lungo la strada. Grazie soprattutto per aver messo prima di ogni cosa, il mio benessere e quello di tutta la famiglia.

A mia madre, Lucia. Ti ringrazio per essermi stata vicino nei momenti più bui, quando temevo di aver perso tutto. Grazie per aver sopportato tutte le mie crisi, per avermi dato la forza di rialzarmi e credere in me stesso. Il nostro è un legame speciale e senza di te oggi non sarei qui.

Alle mie sorelle, Giulia e Valeria, le mie prime sostenitrici. Giulia, grazie per essere stata presente in ogni occasione, nonostante le distanza, per aver appoggiato ogni mia follia e per avermi fatto sentire il migliore in quello che stavo facendo.

Valeria, la mia dose di felicità quotidiana. Grazie per aver reso più leggero questo percorso, per aver vissuto con me ogni sconfitta e ogni traguardo, e per l'amore che mi dimostri ogni giorno.

Rino, Lucia, Giulia e Valeria se ho avuto il coraggio di continuare e non mollare è merito vostro, vi ringrazio.

Ziulvy, presenza fondamentale nella mia vita. Grazie per i tuoi messaggi quotidiani, per i mille gesti e per il supporto che mi hai dato in tutti questi anni.

Zio Vincenzo, zio compare, ti ringrazio per i momenti di spensieratezza, per essere stato partecipe anche con i chilometri che ci dividono.
Voglio dedicare questo traguardo alle mie nonne,

la nonna Ambrosio, il mio angelo custode prima di ogni esame, ovunque tu sia spero spero di averti resa orgogiosa.

la nonna Sabatino, grazie per il tuo infinito affetto e per aver creduto in me; aspettavi questo giorno da tempo e sono felice che oggi tu sia qui a gioire con me.

Vorrei ringraziare chi, questo percorso, anzi questa battaglia, l'ha vissuta in prima linea insieme a me: Giuseppe, Michele, Ciro, Antonio e Ludovica.

Giuseppe, grazie per il tuo sostegno nei momenti più difficili, per aver ascoltato tutti i miei sfoghi e per aver vissuto con me dolori e gioie.

Michele, grazie per le tue frasi prima di ogni esame che hanno contribuito a smorzare la mia costante ansia.

Ciro, il mio primo compagno di uni, una guida essenziale all'inizio del mio percorso che mi ha permesso di superare i primi grandi ostacoli, grazie per avermi aiutato nei periodi più intensi e complicati.

Antonio, senza di te il gruppo non sarebbe mai nato, grazie per aver reso meno noiosi questi anni. Grazie inoltre per le stampe 3D, le docce calde gratis in piscina sono state fondamentali durante la sessione.

Ludovica, una vera amica oltre che collega, grazie per i consigli e per aver tifato per me prima di ogni esame. Grazie per aver compreso i miei stati d'animo e sopportato le mie lamentele quotidiane.

Marta, l'amica con cui ho iniziato, con la quale ho vissuto gli anni più belli e duri dell'università. Ti ringrazio per le nottate a ripetere prima di un esame, per tutti gli inciuci in classe, per i suggerimenti durante gli esami, per aver ascoltato sempre tutti i miei problemi e per essere diventata una presenza costante di cui potermi fidare.

Ringrazio la mia seconda famiglia, i miei amici, grazie per essere stati sempre al mio fianco, per le risate e per l'affetto che mi avete trasmesso nel corso di questi anni.

Nello, grazie perché sei da sempre la persona con cui posso parlare di tutto e condividere le mie paure e i miei sogni. Nel periodo universitario non abbiamo vissuto la solita quotidianità ma, nonostante questo, la tua presenza è stata fondamentale.

Rosario, grazie per essere stato la persona che mi sopporta fin dall'asilo. Tutti i tuoi 'magnatill', prima di un esame mi hanno sempre dato una carica enorme. Grazie inoltre per i tuoi consigli, a volte severi, che mi hanno aiutato a rimettere tutto nella giusta prospettiva.

Raffaele, grazie per avermi assecondato in ogni mia stronzata e per aver alleggerito anche le giornate più tristi, ma soprattutto ti ringrazio per avermi protetto sempre, da ogni punto di vista.

Michele, grazie per gli anni di uni vissuti insieme, senza i quali non avrei frequentato neanche un giorno; sapere di tornare a casa con te dopo un'intensa giornata era la cosa più bella e rilassante.

Ringrazio Peppe e Saba, per le nostre chiacchierate e discussioni sull'informatica, quelle che mi hanno permesso di aprire gli occhi e capire che il percorso che avevo scelto era veramente quello giusto.

Infine ringrazio Alessandra, con cui condivido molti aspetti del mio carattere, per essermi stata vicino e per avermi ascoltato, soprattutto alla fine di questo percorso.

Ci tengo a ringraziare tutti coloro che non ho menzionato ma che ho avuto il piacere di conoscere durante questo percorso.