Analisi, Rappresentazione grafica e salvataggio di dati provenienti da accelerometri

Programmazione e amministrazione di sistema

Mauro Manfredelli 781266

20 Aprile 2016

Indice relazione:

* Introduzione
* Server
* Ricezione dei dati
* Analisi
* Salvataggio
* Rappresentazione grafica

Introduzione

Il progetto richiede la simulazione di una operazione di rappresentazione grafica, analisi e salvataggio di dati provenienti da più accelerometri indossati da una persona.

I dati sono forniti su file, sono dati ottenuti da casi reali registrati usando un set di 5 sensori inerziali XSens e sono riferiti a campionamenti utilizzati per motivi di ricerca. La maggior parte dei campionamenti sono semplici si riferiscono a azioni ripetute, come alzare e abbassare più volte un braccio, o flettere più volte la testa a destra o a sinistra. Altri ancora si riferiscono ad azioni elementari isolate, come sedersi su una sedia, saltare, salire le scale, correre, cadere. Altri prevedono attività semi-naturali di vita quotidiana (Activity of Daily Living: ADL) eseguiti in sequenza come nel caso seguente: il soggetto è inizialmente fermo su una sedia, si alza, cammina per alcuni passi in linea retta, si volta su se stesso di 180 gradi, torna indietro in linea retta e si risiede. Altri campionamenti si riferiscono ad attività di tipo sportivo semplice, come eseguire un calcio karate frontale, o un calcio di karate laterale; altre acquisizioni invece riguardano attività sportive complesse, composte da sequenze di azioni semplici relative all’andare a cavallo.

Le tre componenti principali dell’architettura sono: un set di sensori Mtx per la cattura delle informazioni (ciascuno dotati di accelerometri, giroscopi e magnetometri triassiali) connesse ai movimenti della persona, un palmare in grado di elaborare preliminarmente questi dati e di trasportarli via Wifi, infine, un server che memorizzi questi dati per avere uno storico, che mostri a video e che effettua analisi a runtime e offline dei dati acquisite. Il set di sensori indossabili possiedono un dispositivo bluetooth integrato, che permette di inviare i dati campionati al palmare. Le informazioni che genera il palmare verranno successivamente inviate ad un concentratore e al server attraverso una connessione socket (TCP/IP).

Server

La parte svolta dai sensori inerziali e dal palmare, viene simulata da un applicativo che preleva i dati da un file di log; l’emulatore li invia ogni secondo su una socket TCP/IP riproducendo esattamente quanto capita nel caso reale.

Nel progetto è presente una classe Server che si occupa appunto di accettare le richieste di connessione. La porta su cui si mette in ascolto è la 45555. Il server si occupa di inizializzare i due principali thread dell'applicazione: receiver e analyzer. Questi due thread sono concorrenti e condividono una struttura comune, i campioni. L'accesso ai campioni avviene in mutua esclusione: il receiver scrive e l'analyzer legge. I due thread sembrano partire contemporaneamente, ma in realtà il lavoro dell'analyzer inizia solo dopo aver ricevuto un numero di campioni sufficiente all'avvio dell'analisi. Tale sincronizzazione tra thread avviene mediante dei flag che mi permettono di capire quando iniziare e finire. La funzione di salvataggio dei dati viene svolta dallo steso analyzer. Per comodità durante l'analisi è possibile mettere in pausa il server, riprendere o arrestare l'esecuzione. Il server è gestito da un thread che viene inizializzato nel main, classe Program e gestito tramite metodi statici della classe stessa.

Ricezione dei dati

La classe che si occupa della ricezione è Receiver. E' presente un costruttore che serve per inizializzare la classe dei campioni condivisi e il metodo che si occupa della ricezione dei dati e scrittura degli stessi.

Dal simulatore è possibile scegliere la frequenza di campionamento dei dati: 50, 100 o 200 Hz. Un singolo campionamento di un singolo sensore è pari a 58 byte di dati. Selezionando quindi una frequenza di 50, 100 o 200 Hz sull’emulatore vengono inviati ad ogni secondo 2900, 5800, 11600 byte. I 58 byte di un campionamento sono così suddivisi: 3 byte necessari alla comunicazione (da scartare), 2 byte occupati da un contatore + 52 byte di dati (13 campi da 4 byte) + 1 byte per il CRC, per garantire l'integrità dei pacchetti scambiati. Al momento della connessione l’emulatore invierà come preambolo 14 byte così divisi: 10 byte che rappresentano ID di 10 caratteri che identificano il dispositivo trasmittente; 4 byte che rappresentano la frequenza emulata (la frequenza emulata non corrisponde alla frequenza di campionamento dei file, se un file è stato campionato a 50 Hz e viene letto a 200 Hz i dati saranno solo spediti ad una velocità maggiore). La prima cosa che farà questa classe sarà dunque salvarsi l'id e la frequenza emulata.

Per poter effettuare la ricezione dei dati sono stati associati alla socket un NetworkStream e un BinaryRead. Con il metodo ReadBytes (bloccante) siamo certi di ricevere il numero di byte indicato. Con due cicli, uno per il numero di sensori e uno per i tredici valori, tutti i byte del pacchetto vengono tradotti in valori float, complemento a due, singola precisione (secondo lo standard IEEE 754), grazie al metodo BitConverter.ToSingle(). Tali dati dovranno poi essere salvati su un file .csv. Con il simulatore è possibile avere un numero di sensori da 1 fino a un massimo di 10.

Analisi

Analyzer è la classe che contiene tutta la logica dell'applicazione; si occupa dell'invio dei dati alla finestra di rappresentazione grafica, dell'interpretazione dei dati ricevuti, delle principali funzioni di calcolo e del salvataggio formato .csv dei dati.

Questa classe analizza 500 campioni alla volta. Ogni volta che viene iniziata un'analisi la finestra viene pulita e preparata a una nuova rappresentazione. I dati sono salvati in un vettore tridimensionale sampwin[ , , ] dove il primo indice va da 1 a 'numero di campioni' (massimo 500 per grandi acquisizioni), il secondo da 0 a 4 (numero di sensori massimo 5) e il terzo da 0 a 12 perché 3 valori si riferiscono all'accelerometro, 3 al giroscopio, 3 al magnetometro (x, y, z) e 4 ai quaternioni. I quaternioni li ho inseriti per il calcolo degli angoli di Eulero. Il numero effettivo di campioni lo salvo in una variabile N e la userò per ciclare sui dati; in molti casi ho preferito il 'for each' per evitare la generazione di IndexOutOfRangeException. Il valore del sensore, secondo indice, è fisso e può essere cambiato durante l'esecuzione: possono essere effettuate tre scelte diverse per accelerometro, giroscopio e magnetometro. L'analisi inizia solo quando ho ricevuto la prima parte di dati da analizzare (il Receiver sveglia l'Analyzer).

La prima operazione che viene svolta è il salvataggio dei dati in sampwin; poi viene eseguito l'aggiornamento dei dati nel file .csv (se è la prima volta che lo faccio creo il file); infine viene eseguita l'analisi dei dati ricevuti. Queste operazioni vengono ripetute finchè anche l'ultima finestra è stata analizzata completamente.

La sincronizzazione dei thread di ricezione e analisi avviene anche tramite l'aiuto di una variabile intera 'start': se start=0 non ho nessun invio, se start=1 ho fatto il primo invio e devo creare il file .csv per il salvataggio, se start=2 ho fatto un invio e ho già creato il file .csv. Questa variabile è contenuta nella finestra di dati comuni 'Campioni'.

In questa classe vengono lanciati altri quattro thread utili per la stampa dei Log di analisi riguardo lo stazionamento, il posizionamento, la girata e gli angoli di Eulero. Entriamo nel dettaglio di questa classe:

Calcolo del modulo:

Questo metodo viene usato per calcolare il modulo dei 3 valori dell'accelerometro e dei 3 valori del giroscopio, per poi poter disegnare tale informazioni. E' un banale calcolo del modulo mediante la formula sqrt( x2 + y2 + z2).

Operazione di smoothing:

Serve, per esempio, per eliminare il rumore; dati N campioni, ci si posiziona sul campione i-esimo. Una volta posizionati sul campione i-esimo si prende la finestra di K valori prima e dopo i (ad esempi i 10 campioni prima e i 10 campioni dopo), si calcola il valore medio di questa finestra di 2K+1 valori, e lo si scrive in un vettore smooth[] nel posto i. Si incrementa l’indice i e si procede ripetutamente, fino a quando i = N. Alla fine avremo un vettore di N valori “mediati”, quindi meno sensibili alle micro-variazioni. L'utilizzo di questa operazione durante l'analisi è a completa discrezione dell'utente che può scegliere il valore di k e decidere tramite checkbox se abilitarla o meno.

Calcolo della derivata:

Questa funzione viene fatta tramite RiFunc che calcola il rapporto incrementale. Prende in ingresso un vettore di dati float di una certa dimensione e salva per ogni valore i-esimo il rapporto incrementale tra lui e il successivo. Per semplicità considero un incremento di 1 in modo da non avere nessun denominatore nel calcolo. Questa funzione è stata implementata, ma non viene utilizzata.

Calcolo della deviazione standard:

Ho diviso la logica di questa funzione in due parti: la prima considera una finestra mobile e media fissa, la seconda una finestra

mobile con media mobile. Il funzionamento è praticamente lo stesso solo che nel secondo caso viene usata la funzione smooth per ottenere la media mobile per ogni posizione i del vettore (come valore di k si prende lo stesso valore scelto per il calcolo dello scarto). A seconda dell'indice k scelto viene salvato in un vettore di float la deviazione standard rispetto ai k valori prima, il valore i e i k valori dopo. Questa funzione viene usata per il riconoscimento dello stazionamento (vedi operazioni successive).

Calcolo degli angoli d'Eulero:

Questa operazione viene fatta tramite l'utilizzo dei valori contenuti nei 4 quaternioni alla fine. Si chiamo angoli di Eulero gli angoli che il sistema di riferimento S, solidale col sensore, forma rispetto al sistema geo-referenziato G che è inerziale (in altre parole, è fermo). Questi angoli ci consentono, in concreto, di stabilire di quanto si sia inclinato il sensore nello spazio nelle tre dimensioni. . Dato che lo spazio è tridimensionale i gradi di libertà sono tre, Roll (i.e. φ), Pitch (i.e. θ), Yaw (i.e. ψ). Il valore restituito è in radianti, dimensione per dimensione.

Operazioni per la stampa nei Log:

Ci sono quattro operazioni di stampa. La prima ci permette di stampare lo stazionamento del corpo. Abbiamo due casi: fermo e non fermo (in movimento). Questa stampa avviene mediante l'uso della funzione del calcolo della deviazione standard: se lo scarto è minore di 1 il soggetto è fermo, altrimenti è in movimento; Se lo scarto è compreso tra 0,7 e 1,3 salvo lo stato precedente. Come media setto 9.81 visto che è sempre presente l'accelerazione gravitazionale. Alla funzione del calcolo della deviazione standard passo il vettore contenente i valori ottenuti dal modulo dell'accelerometro. La seconda ci permette di stampare il posizionamento lay, lay/sit, sit e stand. Uso il valore contenuto in x dell'accelerometro del sensore 0 e mi creo il vettore di dati da elaborare. Filtro i valori per un valore k=10 e effettuo lo studio del posizionamento secondo il seguente standard:

X < 2.7 Lay  
2.7 < X < 3.7 Lay / Sit  
3.7 < X < 7 Sit  
X > 7 Stand  
La terza ci permette di stampare di quanto una persona gira a destra o a sinistra. Prima di tutto mi devo calcolare l'angolo che il sensore forma rispetto all'asse x – z mediante la formula arctan( y / z). Calcolo questi valori, angoli in radianti, e li salvo in un vettore. Questi valori saranno utili per la rappresentazione grafica del magnetometro. Non posso usare questi valori cosi come sono per capire se una persona sta girando a destra o a sinistra, ma devo prima filtrarli scartando valori che differiscono di una differenza trascurabile rispetto al successivo. Una volta fatto questo posso analizzarli per capire se una persona sta girando di un angolo significativo (maggiore di 10 gradi). Per capirlo mi baso sulle differenze tra i diversi angoli: se gli angoli sono minori rispetto ai precedenti sto girando a sinistra altrimenti sto girando a destra. Per stampare anche l'ultima girata fatta inserisco un metodo privato separato.  
L'ultima ci permette di stampare gli angoli d'Eulero. Non ho sviluppato nessuna funzione particolare che preveda l'uso di questi angoli, cosi ho deciso di stamparne il valore in un Log ogni 100 campioni analizzati.

Salvataggio

Come già accennato il salvataggio viene fatto dal thread di analisi.  
Une delle funzioni principali di questa applicazione è il salvataggio dei dati ricevuti in formato csv. Al primo campione analizzato viene creato un file csv e, mediante uno stream di scrittura, viene riempito mano a mano con i dati che si ricevono.  
E' presente un metodo dedicato aggiornaCsv che scrive questi dati; essendoci problemi legati alla dimensione del campione ho deciso di effettuare un controllo sul null, ossia 0 essendo float: se il dato che sto guardando è uguale a 0 non lo scrivo altrimenti lo salvo. Viene scritta una riga per ogni campione ricevuto, un blocco per ogni sensore. Alla fine dell'analisi, segnalata mediante una finestra di dialogo separata, sarà possibile trovare il file nella cartella del progetto in bin -> debug.

Rappresentazione grafica

Ho deciso di rappresentare quattro grafici: modulo accelerometro, modulo giroscopio, orientamento magnetometro e posizionamento (lay – sit –stand).

L'analizzatore mediante l'operazione di aggiornamento dei grafici passa alla Form i dati da disegnare. La Form è formata da un pannello di controllo diviso in tre parti: la prima contiene i Log su cui l'analizzatore va a scrivere, la seconda il menu si scelta dello smooth e la terza consente la scelta del sensore da guardare per effettuare l'analisi; a destra delle form è presente il pannello di disegno, dove, in colonna, è possibile vedere i grafici; in alto è presente il tasto Manage che permette l'accesso a una lista di operazioni che permettono di agire sul server: pausa, riprendi, ferma, riavvia.

La parte di modellazione della form è contenuta in MyForm.Designer dove sono presenti le diverse parti grafiche e come sono posizionate nello spazio. Il posizionamento è statico quindi è meglio non toccare le coordinate. Le funzioni principali del della classe MyForm sono createChart e writeChart in quanto l'utilizzo principale della finestra è la creazione e l'aggiornamento dei grafici che viene gestita da questi due metodi. Qualsiasi cosa fatta dall'analizzatore e che bisogna scrivere sulla finestra passa da questa classe; MyForm è formata da molti metodi ma quasi tutti semplici che servono per la gestione degli eventi generati dall'utente (premendo pausa ad esempio) o per la lettura dei dati inseriti dall'utente (per esempio lo smooth o i sensori). È presente anche una parte che gestisce la stampa in una finestra di dialogo di messaggi di cortesia per l'utente. La parte più interessante di questa classe sono i delegate usati per aggiornare i Log o le label; vengono usati se è necessario un metodo Invoke per eseguire chiamate di metodo al controllo, in quanto si trova in un thread diverso da quello in cui è stato creato il controllo.