Sistema di rilevamento della presenza di pedoni sulla carreggiata

Simone Bertolini, Gianmarco Santini,
University of Bologna, Italy

Emails: simone.bertolini@studio.unibo.it, gianmarco.santini@studio.unibo.it

***Abstract*—La mortalitá stradale é un problema molto importante nella nostra comunitá, ogni anno per le strade ci sono circa 3.283 vittime (morti entro il 30 giorno) e 249.175 feriti. Una delle cause principali é sicuramente la distrazione da parte di chi guida. Noi proponiamo un sistema di rilevamento di collisione in grado di avvisare il conducente della presenza di un pedone che gli attraverserá la strada. Tale sistema é soggetto a vincoli importanti in quanto la localizzazione di una persona puó richiedere un consumo di energia proibitivo da parte dello smartphone e il raggio di rilevamento dei dispositivi é limitata. Il nostro sistema é pensato come un'estensione per un sistema di navigazione come google maps in modo da poter avvisare il conducente direttamente sulla mappa dove si troverá il pedone. Come verrá descritto in seguito tale sistema comprende due modalitá di funzionamento la prima dedicata al pedone e la seconda dedicata al conducente di un veicolo. La comunicazione tra i vari dispositivi avviene tramite Wi-Fi Direct e la localizzazione del pedone avviene tramite l'uso congiunto di GPS e i sensori dello smartphone.**

I. Introduzione

Una delle cause principali di mortalitá in italia sono sicuramente gli incidenti stradali, questi ogni anno mietono migliaia di persone e costituiscono sicuramente un problema importante. Nel 2017 sono stati 174.933 gli incidenti stradali con lesioni a persone, con 3.378 vittime (morti entro 30 giorni dall'evento) e 246.750 feriti. Tra questi sono presenti i pedoni investiti da automibilisti distratti alla guida che non si accorgono della loro presenza finché non é troppo tardi. Questo é il problema che il nostro sistema cerca di risolvere, di seguito nel capitolo II vengono descritte tecnologie e metodi che hanno contribuito alla pianificazione e sviluppo del sistema, nel capitolo III viene descritta l'architettura dell'applicazione e il funzionamento, nel capitolo IV viene descritta l'implementazione, nel capitolo V vengono analizzate le performance e infine nel capitolo VI vengono tratte le nostre conclusioni. (da togliere poi)Descrivi qui il contesto generale, i contributi del progetto, e la struttura del documento.

II. Related works

In questa sezione vengono descritti metodologie e strutture che hanno aiutato allo sviluppo del sistema. [1] Descrive un sistema a basso costo e consumo per la localizzazione continua di un dispositivo con buona accuratezza, questo sistema fa uso di GPS in concomitanza con accelerometro, giroscopio e magnetometro in modo da ridurre i consumi sui dispositivi interessati. Tale sistema peró é dedicato a una localizzazione della persona non solo all'esterno ma anche all'interno di edifici, mentre il nostro si occupa solo della localizzazione all'esterno. [2] Descrive un'algoritmo per tracciare e localizzare un veicolo in tempo reale utilizzando GPS, accelerometro e giroscopio su uno smartphone con sistema android. Tale algoritmo in grado rispondere ai cambi di velocitá e di direzione in tempo reale senza avere informazione dal GPS. [3] Illustra un sistema per il tracciamento di percorsi di guida ibrido che utilizza barometro, GPS, accelerometro e giroscopio cercando di ridurre i consumi energetici dello smartphone. Tale sistema é pensato per misurare gli spostamenti dello smartphone in modo continuo e per identificare certi spostamenti conosciuti che le persone ripetono pi spesso. Infine [4] descrive in modo completo il funzionamento e l'implementazione dell'architettura Wi-Fi direct che come verrá descritto successivamente é quella che é stata utilizzata per lo sviluppo del nostro sistema. Fornisci una breve rassegna di articoli di ricerca, software, prototipi o tecnologie che sono collegate in qualche modo al problema affrontato nel progetto. Tutti i lavori devono essere referenziati ed inseriti nella Bibliografia.

III. Architettura

Il sistema é stato progettato per funzionare su ambiente Android, permetttendo in questo modo di essere utilizzato in larga scala e senza costi aggiuntivi al prezzo del dispositivo stesso. Tale sistema é costituito da due tipologie di layer applicativi che potremmo denominare come Publisher e Observer, cosí da poter rendere il piú intuitivo possibile il ruolo da loro ricoperto.

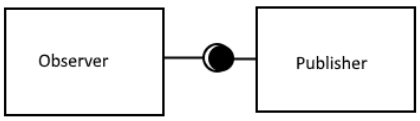

Fig. 1. Concetti base

Il ruolo del Publisher viene assunto da tutti i dispositivi in possesso ai pedoni, il loro compito principale é quello di avvisare i veicoli circostanti riguardo la propria posizione corrente e la direzione in cui si stanno muovendo. Tutti i veicoli presenti sulla strada, invece, assumono il ruolo di Observer ed hanno il compito di ascoltare in ogni istante

i Publisher, elaborare i loro dati e avvertire i conducenti in caso di pericolo. La struttura della rete risulta dunque essere un insieme approsimatamente omogeneo di Publisher ed Observer. La forza di questo sistema il fatto che la rete non preveda una vera e propria sturuttra, ma che si adatti a quelle che sono le limitazioni dell'ambiente circostante. Questa qualit del sistema lo rende flessibile e particolarmente scalabile.

IV. Implementazione

L'applicazione é divisa in due modalitá, Publisher e quella Observer. La prima é quella relativa al pedone che si occupa di tracciare i suoi movimenti, la sua velocitá e la sua direzione di movimento. La seconda quella del veicolo che si occupa di elaborare le informazione fornite dai pedoni e stimare in base ad esse se sará presente una collisione. Di seguito verranno descritte le informazioni e i vari passaggi che hanno portato allo sviluppo dell'applicazione.

A. Ambiente di Sviluppo

Come giá descritto in precedenza, il sistema é stato completamente sviluppato per sistema operativo Android in modo da consentire la maggior portabilitá del software. Infatti il nostro obiettivo non era solo la verifica verticale delle funzionalitá del software, ma anche il confronto orizzontale tra dispositivi che utilizzano hardware differenti. Questo ha consentito di trovare i punti in comune ma anche le discrepanze tra i diversi dispositivi hardware e di generare un software flessibile in grado di adattarsi al dispositivo host. Per lo sviluppo é stato utilizzato l'IDE di default di Android ovvero Android Studio con linguaggio di programmazione Java.

B. Obiettivi

L'idea iniziale era quella di riuscire ad inidividuare un pedone con il maggior preavviso possibile, ma limitando il consumo energetico del dispositivo ad egli in possesso. Per fare ció abbiamo effettuato alcuni test utilizzando il Bluetooth, la tecnologia a radio frequenza con il miglior risparmio energetico. In particolare ci siamo concentrati sull'utilizzo dei Beacon Bluetooth inviati dal dispositivo pedone ed elaborati dal veicolo. Al loro interno sono stati inseriti dati fittizi che simulassero un payload di dimensioni verosimili. Le prove effettuate includevano l'invio dei beacon sia dal veicolo in movimento che dal pedone a velocitá ridotta. Purtroppo in entrambi casi i risulati ottenuti sono notevolmente scadenti. Infatti i dispositivi riuscivano a trovarsi solo a distanze quasi nulle (non piú di una decina di metri). La nostra seconda scelta á dunque ricaduta sull'utilizzo del WiFi Direct(P2P). Nel paragrafo seguente vedremo in breve come funziona questo standard.

C. WiFi Direct e Problemi riscontrati

Wi-Fi Direct (nato con il nome Wi-Fi P2P - Wi-Fi Peer-to-Peer) é uno standard della Wi-Fi Alliance, ormai comune in molti prodotti come smartphone, tablet, stampanti, video-camere digitali, console di gioco e scanner, che consente a due o pi dispositivi certificati di connettersi tra loro con un collegamento Wi-Fi diretto senza aver bisogno di un router wireless o di un hotspot. In particolare questa tecnologia prevede 3 fasi prima dio poter scambiare dati:
1) Effettuare il discovery dei dispositivi circastanti
2) Creazione di un gruppo (se questo non esiste giá) al fine di poter poi accettare la connesione
3) Effettuare handshake di connessione

Come racconta lo standar IEEE, il WiFi P2P garantisce una riuscita della connessione all'interno del range dei 13 secondi. Purtroppo, per il nostro applicativo queste latenze sarebbero state enormi ed avrebbero reso il sistema inutilizzabile. Abbiamo dunque deciso di sviluppare un applicazione di chat base, che ci aiutasse a stimare il reale tempo di riuscita della connessione tra due dispositivi. I test ci hanno riportato quanto segue:
- Tempo di connessione 3/4,
- Una volta stabilita la connessione il tempo di risposta ad un messaggio era pressoché nullo. I 3-4 di connessione non rispecchiavano in tutto per tutto le nostre esigenze in quanto, non certi del range del segnale WiFi, 3s * (50km/h / 3.6) = 40m di range sarebbero stati persi in partenza.

Non potendo consentire la connessione tra i dispositivi, ci rimaneva unicamente la fase di Discovery da sfruttare. All'interno del record di Discovery, infatti, vi si puó inserire qualunque dato sottoforma di tabella hash ed il processo di segnalazione e di ricezione del segnale é pressoché immediato. Fatte queste considerazioni, abbiamo deciso di demolire l'applicazione di chat lasciando unicamente la parte di discovery ed effettuare svariati test in cui misuravamano la distanza minima in cui i due dispositivi riuscivano a vedersi per la prima volta. I riusultati sono stati eccezionali. Infatti i valori in metri ottenuti sono stati da un centinaio di metri in campo aperto a 45-50 metri in curva con traiettoria in linea d'aria offuscata da ostacoli. Questo principio é risultato per noi essere le fondamenta vere e proprie del sistema stesso. Sfortunatamente le librerie di android non funzionano molto bene per quanto riguarda il publishing ed il discovery nel WiFi Direct. Infatti se si vuol pubblicare i propri dati sulla rete bisogna anche effettuare un'operazione continua di Discovery, altrimenti nessuno dato viene spedito. Inoltre (considerando che il Discovery chiama una callback ogni volta che trova un dispoositivo) la callback del Discovery non segnala lo stesso dispositivo piú di una volta se esso é giá stato trovato anche se possiede un record differente. Questo ci ha costretti a resettare la procedura del discovery ogni T secondi in modo da poter mettere il dispositivo Observer nuovamente in ascolto anche di Pubblisher giá trovati. Per valori di T inferiori ai 7-8 secondi si rischia di mandare in crash il driver del modulo WiFi che, non so se per problemi interni o per verifiche di funzionamento del sistema operativo, viene disattivato bloccando qualsiasi forma di ricezione e/o trasmissione. L'applicativo lato publisher fornisce all'interno del proprio record di service discovery i seguenti dati che varrannoo poi discussi nei paragrafici successivi nei relativi modelli.
- Longitudine

- Latitudine
- Velocita
- Timestamp
- Timestamp della Posizione
- Accuratezza
- Bearing

Lato Observer invece, i dati di tutti i pedoni trovati tramite il discovery vengono salvati (eventualmente aggiornati se gia trovati in precedenza), processati ed alla fine del loro ciclo vita eliminati.

D. Test Effettuati

Per quanto riguarda la localizzazione del pedone l'idea é stata quella di usare il GPS unito ai sensori dello smartphone in modo da ottentere una localizzazione abbastanza precisa con un consumo energetico contenuto. A questo proposito é stato necessario eseguire qualche test per controllare quali output i vari sensori fornissero e come utilizzare quest'ultimi. In particolare é stato necessario suffermarsi sul rotation vector, in quanto ci serviva la direzione in cui il pedone sta camminando, per controllare quanto preciso era il calcolo dell'azimuth, l'angolo relativo al nord magnetico. Quest'angolo é risultato essere molto sensibile cioé, il valore non rimaneva fisso ma tendeva a cambiare al minimo movimento dello smartphone. Altri test sono stati eseguiti con lo step counter, il sensore che dovrebbe segnalare ogni volta che viene effettuato un passo, esso é risultato essere un pó inpreciso nei passi corti e nei cambi di direzione dove a volte segnava dei passi non effettuati. Inoltre le segnalazione della presenza di un passo non era immediata ma poteva accadere che venisse segnalata dopo aver effettuato un nuovo passo indicando la presenza di entrambi.

E. Modello di Localizzazione

Di seguito viene descritto come avviene la localizzazione del pedone e come vengono calcolati i vari dati da inviare tramite Wi-Fi direct ai vari veicoli presenti nel range di trasmissione. La classe che si occupa di fare ció é la classe HumanLocalService, tale classe sfrutta i valori dei sensori rotation vector e step detector per stimare la posizione del pedone. Al"interno del costruttore infatti vengono implementati i vari listener che si occupano di eseguire delle operazione nel momento in cui avviene un aggiornamento nei dati rilevati dai sensori. Qui sotto é possibile osservare il codice relativo al listener del sensore rotaion vector, dai dati restituiti dal sensore si ricava la matrice di rotazione del dispositivo, una matrice 4 x 4, attraverso la funzione "getRotationMatrixFromVector". Una volta ottenuta la matrice viene effettuato un cambio di coordinate in base a come é orientato il dispositivo, questo avviene con la funzione "remapCoordinateSystem", nel nostro caso il device viene considerato sempre in portrait mode quindi gli assi da usare sono X e Z positive. Infine data la nuova matrice di rotazione nelle coordinate giuste attraverso la funzione "getOrientation" vengono calcolati azimuth, pitch e roll del dispositivo. Come spiegato in precedenza i valori ottenuti sono molto sensibili quindi é stato inserito un offset, di 10 gradi circa, in modo che il valore dell'angolo rimanga inalterato a meno di una variazione maggiore dell'offset.

```
public void onSensorChanged(SensorEvent
    sensorEvent) {
   float offset = (float)Math.toRadians(10);
   float[] rotationMatrix = new float[16];
   SensorManager.getRotationMatrixFromVector(
   rotationMatrix, sensorEvent.values);
   float[] remappedRotationMatrix =
   new float[16];
   SensorManager.remapCoordinateSystem(
         rotationMatrix,SensorManager.AXIS_X,
         SensorManager.AXIS_Z,
         remappedRotationMatrix);
   float[] orientations = new float[4];
   SensorManager.getOrientation(
         remappedRotationMatrix,
         orientations);
   angle = orientations;
   if (angolo == 0) angolo = angle[0];
   else if (Math.abs(angolo-angle[0])>offset)
       angolo = angle[0];
   }
```

Ricordiamo che l'azimuth é l'angolo rispetto al nord magnetico mentre il pitch e il roll sono rotazioni del device rispetto alle assi X e Y come si puó vedere in fig 2. L'altro sensore

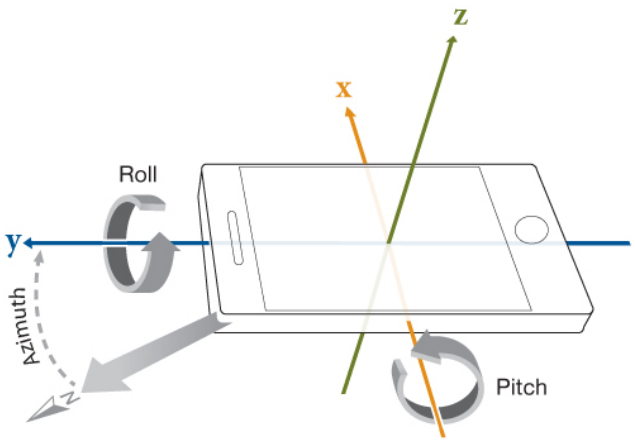

Fig. 2. Azimuth Pitch e Roll

utilizzato é lo step detector, quest'ultimo informa quando avviene un passo permettendo insieme all'azimuth ottenuto di stimare la posizione del pedone. Di seguito si pu vedere il codice relativo al listener che andremo a descrivere ora. Ogni volta che viene rilevato un passo vengono incrementati i valori di step e step2, questi sono dure contatori che servono rispettivamente per il calcolo dei passi prima di richiedere un aggiornamento al GPS e prima di calcolare la velocitá, inoltre viene salvato il timestamp di quando é avvenuto l'ultimo passo. La velocitá viene calcolata ogni cinque passi in quanto il listener non informa dell'avvenimento di un passo in modo immediato ma lo fa in modo non regolare a volte, in questo modo abbiamo un valore approssimato che si avvicina di piú

a quello reale. Tale calcolo avviene tramite la formula fisica velocitá = spazio / tempo, lo spazio é dato dal numero dei passi, nel nostro caso cinque, moltiplicato per la lunghezza del passo, questa lunghezza viene calcolta nel momento in cui viene instaziato l'oggetto della classe HumanLocalService usando il valore dell'altezza dell'utente richiesta in input tramite un dialog, mentre il tempo viene calcolato come il tempo di quando viene effettuato il quinto passo meno quello di quando é stato effettuato il primo passo. La velocitá che viene restuita infine é il massimo tra il valore calcolato e 1, questo serve per riuscire a prevedere quei casi in cui i pedoni si fermano e poi riprendono a camminare all'improvviso. Oltre a ció viene salvato l'azimuth calcolato nel momento in cui é stato rilevato un passo e vengono calcolate le nuove coordinate attuali in lat e long conoscendo quelle precedenti, l'angolo rispetto al nord e la lunghezza di un passo. Per quanto riguarda la richiesta delle coordinate reali al GPS essa viene fatta in due casi, se sono stati fatti 30 passi oppure se sono passsati 25 secondi dall'ultimo aggiornamento, in questo modo si cerca di mantenere la posizione accurata. La chiamata al GPS avviene tramite "getCurrentLocation" che aggiorna le coordinate attuali a quelle reali ottenute dal gps, mentre la chiamata ogni 25 secondi avvine tramite l'handler "hnd".

```
public void onSensorChanged(SensorEvent
    sensorEvent) {
   step++;
   step2++;
   timestamp = System.currentTimeMillis();
   if (fst == true) {
      startTime = System.nanoTime();
      fst = false;
   }else if(step2%5==0){
      step2 = 0;
      endTime = System.nanoTime();
      double sp = 5*(stepLength / 100) /
          ((endTime - startTime) /
          Math.pow(10, 9));
      speed = Math.max(sp,1);
      startTime=endTime;
   }
   if (angle != null) {
   azimuth = angolo;
   scX+=(getDistance(
   Math.cos(azimuth))/r_earth)*
   (180/Math.PI)/Math.cos(scY*Math.PI/180);
   scY+=(getDistance(
   Math.sin(azimuth))/r_earth)*
   (180/Math.PI);
   if (step%30==0) {
      hnd.removeCallbacks(rn);
      getCurrentLocation();
      hnd.postDelayed(rn, 25000);
      }
   }
}
```

Un altro parametro calcolato é l'accuratezza che viene misurata come la distanza in metri tra la posizione stimata e quella reale ottenuta dal GPS al momento di una chiamata ad esso.

F. Modello di Previsione

Questo modello viene implementato lato Observer e si occupa di elaborare i dati ricevuti dai diversi Publisher ed incrociarli con quelli del veicolo che rappresenta. Un modello di previsione prevede come input il tragitto che il veicolo dovra percorrere sottofroma di Polilinea (vettore di segmenti) ed un gestore di posizionamento che sfrutti il segnale GPS in modo da sapere in ogni istante le cordinate attuali del veicolo. Dati questi input, per ogni Publisher trovato stima se in un tratto della Polilinea di input il veicolo potrebbe andare a collidere con il Publisher. Nel caso lo segnala sulla mappa. La stima di collisione viene effettuata utilizzando il seguente algoritmo.

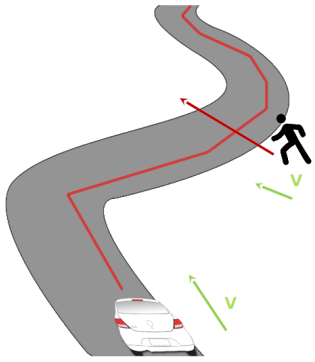

Fig. 3. Modello di Previsione

Il concetto di fondo é che l'uomo colliderá con il veicolo solo se la sua traiettoria e la sua velocitá partendo da un punto A lo porteranno in un punto B tale per cui il segmento AB interseziona uno dei tratti della Polilinea realitiva al tragitto del veicolo. La seguente funzione verifica se due segmenti si intersezionano.

```
/**
* Check if the human will intersect the
    vehicle
* @param h human service Object
* @return True -> Intersection
*/
public boolean
    willHumanIntersectVehicle(HumanService h)
{
  LatLng loc = getCurrentPosition();
  int start = getCurrentPolylineSegment();
  int end = start;
  // Search indexes on which check
     intersection
  for(int i = start; i < route.size()-1; i++)
  {
    double d =
        getPointSegmentProjectionDistance(
      route.get(i),
      route.get(i+1),
      loc);
    if(d < WIFI_MAX_RANGE)
    {
      end = i+1;
```

```
    }else{
      break;
    }
  }
  if(end == start)
  {
    // All segments are inside 200 meters
        range
    end = route.size()-1;
  }

  // Intersection check
  for(int i = start; i < end; i++)
  {
    if(doIntersect(
      route.get(i), // P1 of the vehicle
      route.get(i+1), // P2 of the vehicle
      new LatLng(h.latitude, h.longitude),
          // P1 human
      h.getHumanPositionIn(
      (int) Math.ceil(WIFI_MAX_RANGE /
          getCurrentSpeed())) // P2 of the
          human
    ))
    {
    return true;
    }
  }
  return false;  // No intersections
}
```

Prima di tutto all'interno di questa funzione viene richiesta la posizione corrente del veicolo ed il segmento di polilinea all'interno del quale esso si trova. Per trovare il segmento di polilinea corrente (funzione *getCurrentPolylineSegment*), viene effettuato un calcolo trigonometrico della distanza tra la posizione del veicolo ed ogni segmento. Successivamente il tratto con la distanza punto-segmento minore eńconsiderato quello corrente. La distanza punto-segmento in geometria sferica viene calcolata con il seguente algoritmo. Definiamo con AB il segmento in questione e con C la posizione del veicolo.
1) Convertire le coordinate terrestri in coordinate Cartesiane (utilizzando come orgine il centro della terra).
2) Calcolare T, il punto sulla retta AB piuv́icono al punto C utilizzando i tre seguenti prodotti vettoriali:
a) G = A x B
b) F = C x G
c) T = G x F
3) Normalizzare T a moltiplicarlo per il raggio della terra.
4) Riconvertire il valore in latitudine e lonfitudine.
5) Verificare che il punto si trovi esattamente sul segmento AB, nel caso sovrfascriverlo con l'estremo del segmento piuv́icino.
6) Utilizzare la formula della distanza tra due punti in geometria sferica per trovare la distanza punto-segmento.

Proseguento con l'analisi della funzione *willHumanIntersectVehicle*, il primo ciclo for, partendo dal segmento corrente, cerca quali sono i segmenti di polilinea la cui distanza tra segmento e posizione del veicolo risulta essere inferiore al range del WiFi, definito dalla costante WIFI_MAX_RANGE. Questo perchesi presume che il veicolo non possa ricevere un segnale piudistante del normale range di trasmissione. Una volta terminata questa verifica, avremo un estremo superiore ed un estremo inferiore che limitano il numero di indici dei segmenti sui quali poi verraéseguito il controllo di intersezione con il segmente pedone. Il secondo ciclo for esegue per ogni segmento compreso nell intervallo prima definito il controllo di intersezione con il segmento del pedone. Questo controllo viene eseguito dalla funzione *doIntersect* alla quale vengono passati i due punti del segmento veicolo, la posizione del pedone descritta all'interno del record di Discovery e la stima della posizione del pedone tra J secondi, dove J eíl tempo impiegato dal veicolo a percorre il range del WiFi. La nuova posione viene calcolata utilizzando un moto rettilineo uniforme dove il tempo eúguale J + timestamp corrent - timestamp_pos ottenuto dal service discovery del pedone. Timestamp_pos eí'istante in cui eśtata calcolata la posizione del pedone prima di essere inviata. *doIntersect* eí'unica funzione ad utilizzare calcoli euclidei invece che sferici. Questo perchela superfice di una sfera per porzioni molto piccole, puoéssere considerata una superficie piana. Il controllo di intersezione viene sviluppato utilizzato il concetto di orientazione tra punti. Due segmenti (p1,q1) e (p2,q2) si intersezionano se e solo se una delle seguenti condizione é verificata.
1. Caso generale:
(p1, q1, p2) e (p1, q1, q2) hanno orientazioni differenti && (p2, q2, p1) e (p2, q2, q1) hanno orientazioni differenti.
2. Caso speciale:
(p1, q1, p2), (p1, q1, q2), (p2, q2, p1), e (p2, q2, q1) sono collineari && le proiezioni in x di (p1, q1) e (p2, q2) si intersezionano && le proiezioni in y di (p1, q1) e (p2, q2) si intersezionano.
L'insieme di tutti questi algoritmi danno vita al modello di previsione.

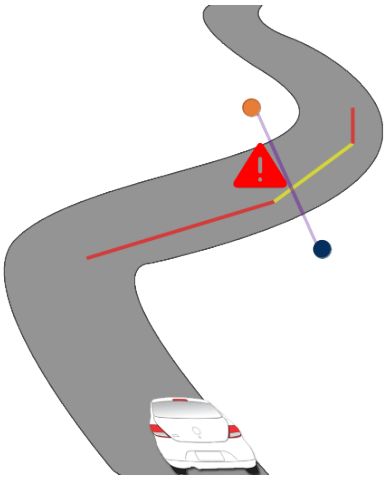

Fig. 4. Intersione tra segmenti

V. Valutazione performance

Sfortunatamente effettuare delle prove con questo sistema non risulta per niente semplice. Questo perchéśarebbe necessario scegliere un'area urbuana con un alto numero di pedoni e veicoli per intravedere tutti i minimi particolari e poter dunque costruire modelli realistici. Detto questo, i nostri test sono stati effettuati in aree rurali (per non intralciare il traffico), con perodiversi ostacoli tra trasmettitore e ricevente. Infatti gli scenarii da noi scelti ed analizzati sono i seguenti.

	Sole		Pioggia	
	$\perp$	//	$\perp$	//
Linea diretta	1	2	3	4
Curva/Ostacoli	9	10	11	12

Velocitá del veicolo $\sim$ *50Km/h*

VI. Conclusioni

Conclusioni, possibili sviluppi futuri e limitazioni del progetto realizzato

References

[1] Subhanjan Saha, Samrat Chatterjee, Amit Kr. Gupta, Indrajit Bhattacharya, Tamal Mondal TrackMe - A Low Power Location Tracking System Using Smart Phone Sensors *Conference on Computing and Network Communications (CoCoNet'15)*,2015.

[2] Siim Plangi Real-time Localisation and Tracking System for Navigation Based on Mobile Multi-sensor Fusion *UNIVERSITY OF TARTU Institute of Computer Science*, 2018.

[3] Myounggyu Won, Member, IEEE, Ashutosh Mishra, Student Member, IEEE, and Sang H. Son, Fellow, IEEE HybridBaro: Mining Driving Routes Using Barometer Sensor of Smartphone *IEEE*, 1558-1748, 2017.

[4] Wi-Fi Alliance. Wi-Fi Peer-to-Peer (P2P) Technical Specification *Wi-Fi Alliance*, 2016.