UNITA' TECNICA TECNOLOGIE DEI MATERIALI BRINDISI (UTTMATB)	*Tipo di documento* **Rapporto tecnico**	*Pagina / di Pagine*	1 / 14
		Sigla di identificaz. Revisione	RT 02/13 rev. 0

TITOLO DOCUMENTO:

RENDERING REMOTO DI FILE PLY GENERATI DA LASER SCANNER 3D SU DISPOSITIVI MOBILI

***SOMMARIO**: Il Laser Scanner tridimensionale è una tecnologia ormai affermata per il rilievo e l'acquisizione della forma di oggetti complessi. I campi di impiego sono i più vari, pensiamo alle applicazioni nel settore industriale (reverse engineering) nel settore medico o nei beni culturali per le ricostruzioni architettoniche e delle opere d'arte. L'obiettivo di questo lavoro è rendere fruibili i modelli tridimensionali ottenibili dalle scansioni laser su dispositivi mobili (smartphone, tablet), sfruttando il rendering remoto su workstation ad alte prestazioni.*

Parole chiave: Dispositivi Mobili, Remote Rendering, Computer Graphics, Client/Server

***ABSTRACT**: The three-dimensional laser scanner is a modern technology used to detect and capture the shape of complex objects. There is a lot of possible application fields for this technology: applications in the industrial sector (reverse engineering), in the medical field or in cultural heritage field for reconstruction of works of art. The main goal of this work is to make accessible the three-dimensional models obtained from laser scans on mobile devices (smartphones, tablets), exploiting a remote rendering application on high-performance workstations.*

Keywords: Mobile Devices, Remote Rendering, Computer Graphics, Client/Server

NOTE: ATTIVITA' SVOLTA NELL'AMBITO DEL PROGETTO IT@CHA PON 2007-2013

AUTORE: Fabio De Pascalis

0		**Fabio De Pascalis**	**Antonella Rizzo**	**Leander Tapfer**	**08/04/2013**
REV.	***Motivo della revisione***	***Redazione***	***Convalida***	***Approvazione***	***Data di emissione***
		(nominativo, data e firma)			

INDICE

1. Introduzione

La rivoluzione tecnologica nella telefonia mobile sta rapidamente trasformando il modo di comunicare e le abitudini di vita di milioni di persone. Potenze di calcolo sempre più elevate, reti ad alta velocità, display ultra brillanti, e a breve, anche flessibili stanno rivoluzionando il concetto di mobilità e di interazione. Le prestazioni di rendering dei nuovi dispositivi mobili stanno vertiginosamente crescendo con l'impiego di schede grafiche ad hoc (NVIDIA, ADRENO), tuttavia non sono chiaramente paragonabili a quelle proposte nelle moderne WORKSTATION GRAFICHE multicore/gpu, che sono ancora l'unico strumento in grado di consentire la visualizzazione di dataset poligonali di dimensioni superiori ad una certa soglia. La soluzione proposta in questo lavoro si basa sull'impiego di un server ad alte prestazioni per il rendering dei dataset 3D e di un client android che gestisce l'interazione con la scena 3D e riceve in streaming in tempo reale le immagini generate dal server.

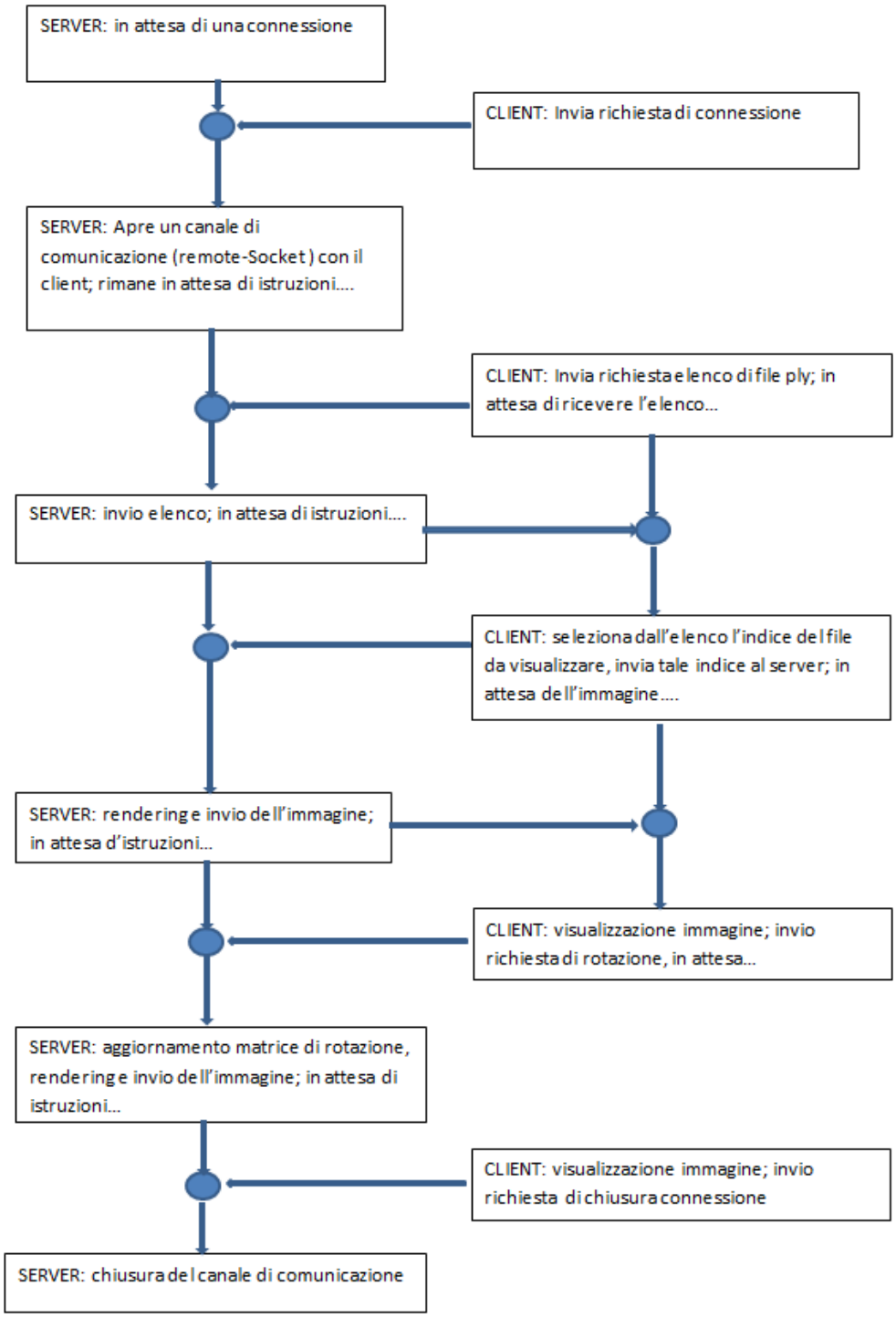
2. Architettura del sistema

Il sistema proposto si basa su un'architettura client/server composta da un client Android per dispositivi mobili che tramite connessione Wireless 802.11 comunica con una workstation 3D (Pc-server).

Il server è stato realizzato in c++, il suo motore di rendering si basa sulle librerie grafiche Vtk (OpenGL) mentre l'interfaccia grafica è stata sviluppata con le WxWidgets. Il compito del server è quello di memorizzare e visualizzare i file ply selezionati dal client, gestire la comunicazione con il client interpretando gli eventi e i comandi che quest'ultimo invia, aggiornare la matrice di trasformazione (Rotazione/traslazione/Scaling) del modello 3D, ed inviare il risultato del rendering (immagini 2D) in formato compresso (jpeg) verso il client.

Il Client è implementato tramite java e le librerie Android, esso consente di connettersi in tcp al server, ricevere la lista dei file ply disponibili, selezionare il file e richiedere al server il rendering dello stesso, gestire l'interazione con l'oggetto 3D, decomprimere e visualizzare in un'apposita finestra l'immagine ricevuta in streaming.

Un tipico diagramma degli eventi che sintetizza il protocollo di comunicazione e lo scambio di messaggi tra client e server è il seguente:



Si noti che il server rimane in un permanente stato di attesa per eventuali richieste di connessione all'interno di un ciclo "idle", mentre una volta accettata la connessione essa

è gestita attraverso l'impostazione di un canale di trasmissione remoto (remotesocket) all'interno di un proprio thread. Questo permette la gestione simultanea ed indipendente (multithreading) di più processi di rendering per un certo numero di utenti client. E' possibile in ogni modo limitare il numero massimo di connessioni in funzione delle caratteristiche e risorse hardware che si hanno a disposizione.

3. Formato File Ply

Il formato file ply (polygonal file format) è un formato ideato presso l'università di Stanford particolarmente adatto a rappresentare oggetti poligonali e dati tridimensionali prodotti da laser scanner 3D. Esso consiste in un header seguito da una lista di vertici e una lista di poligoni. L'header specifica il numero di vertici e di poligoni che compongono il file e le proprietà (tali come coordinate (x,y,z), normali, colori) associate ad ognuno di essi. I poligoni sono definiti specificando un numero "n" che identifica il numero di vertici che lo compongono e da una lista di indici che rappresentano i puntatori all'interno della lista di vertici; di seguito un esempio di file ply in formato ASCII che descrive un quadrato composto da 4 vertici connessi con due triangoli:

```
ply
format ascii 1.0
comment …….
element vertex 4
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
element face 2
property list uchar int vertex_indices
end_header
0 0 0 255 0 0
2 0 0 125 0 0
2 2 0 255 255 0
0 2 0 255 0 255
3 0 1 2
3 2 3 0
```

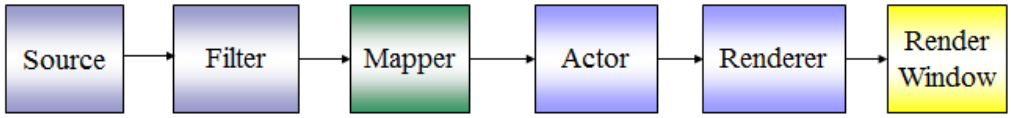
4. Server-VTK

Il sistema di rendering del server è stato sviluppato utilizzando le librerie VTK, acronimo di "The Visualization Toolkit". La scelta di questo strumento è stata determinata dalle sue capacità in termini di visualizzazione scientifica e dalla possibilità di estendere e introdurre nuove funzionalità di base agendo direttamente sul codice sorgente distribuito nella forma "open source" e di realizzare applicazioni cross-platform. VTK comprende un insieme di classi scritte in C++ e si basa sullo standard grafico OpenGL per i processi di visualizzazione.

Vediamo un estratto del codice c++ che mette in evidenze come l'uso delle classi VTK consenta con poche istruzioni di ottenere la visualizzazione di oggetti 3d in formato ply.

```
vtkPlyReader *reader=vtkPLYReader::New();
reader->SetFileName(file);
vtkOpenGLPolyDataMapper *mapPD = vtkOpenGLPolyDataMapper::New();
mapPD->SetInput(reader->GetOutput());
vtkActor *actorPD = vtkActor::New();
actorPD->SetMapper(mapPD);
vtkRenderWindow *renWin = vtkRenderWindow::New();
vtkRenderer *renderer=vtkRenderer::New();
renderer->AddActor(actorPD);
renWin->Render();
```

In pratica i processi di visualizzazione di vtk si basano sull'impostazione di una pipeline grafica che è composta usualmente dai seguenti elementi base:



La classe source generalmente consiste in un file reader o in un generatore di dati; nel nostro caso è un lettore di file PLY (vtkPlyReader) messo a disposizione dalla libreria standard VTK.

Il Filter elabora i dati in uscita dalla sorgente applicando un algoritmo di data-processing (ad esempio un filtraggio); nel nostro caso tale blocco è assente.

Il Mapper (vtkOpenGLPolyDataMapper) è un elemento fondamentale della pipeline, esso traduce la struttura geometrica dei dati in chiamate e funzioni OpenGL determinando le primitive grafiche che attivano il processo di rendering sull'hardware grafico.

L'Actor specifica invece le proprietà dell'oggetto grafico, in termini di posizione, colore, texturing ecc.

Il Render coordina il processo di rendering (OpenGL) includendo nella scena grafica luci, camera e attori.

La RenderWindow rappresenta il contesto grafico (finestra) nel quale visualizzare il risultato della pipeline.

L'elemento vtkOpenGLPolyDataMapper della distribuzione standard vtk consente la visualizzazione e la conseguente generazione di primitive grafiche per qualsiasi topologia di struttura dati poligonale, questa sua genericità penalizza però le prestazioni e la fluidità del rendering. Per questo motivo abbiamo deciso di sostituirla con una classe da essa derivata, vtkOpenGLPlyMapper, che fosse specifica per la visualizzazione di file Ply le cui celle fossero composte solo da elementi triangolari (la maggior parte di quelli da noi trattati). Sostanzialmente la modifica è consistita nel definire un override di alcuni metodi della classe base, tra cui il metodo DrawPolygons. Mentre la classe base genera per ogni cella del file ply una serie di chiamate glVertex(), la classe derivata sfrutta il costrutto OpenGL dei vertex_array sostituendo un'unica primitiva grafica alle n precedenti. Questo si concretizza in un aumento sostanziale del fame rate come si evince dal grafico in fig.1 dove si confrontano le prestazioni delle due classi (le prove sono state condotte su una calcolatore Intel core i3 S30, dotata di 16Gb di RAM e GPU NVIDIA QUADRO 4000, Windows 7 64bit)

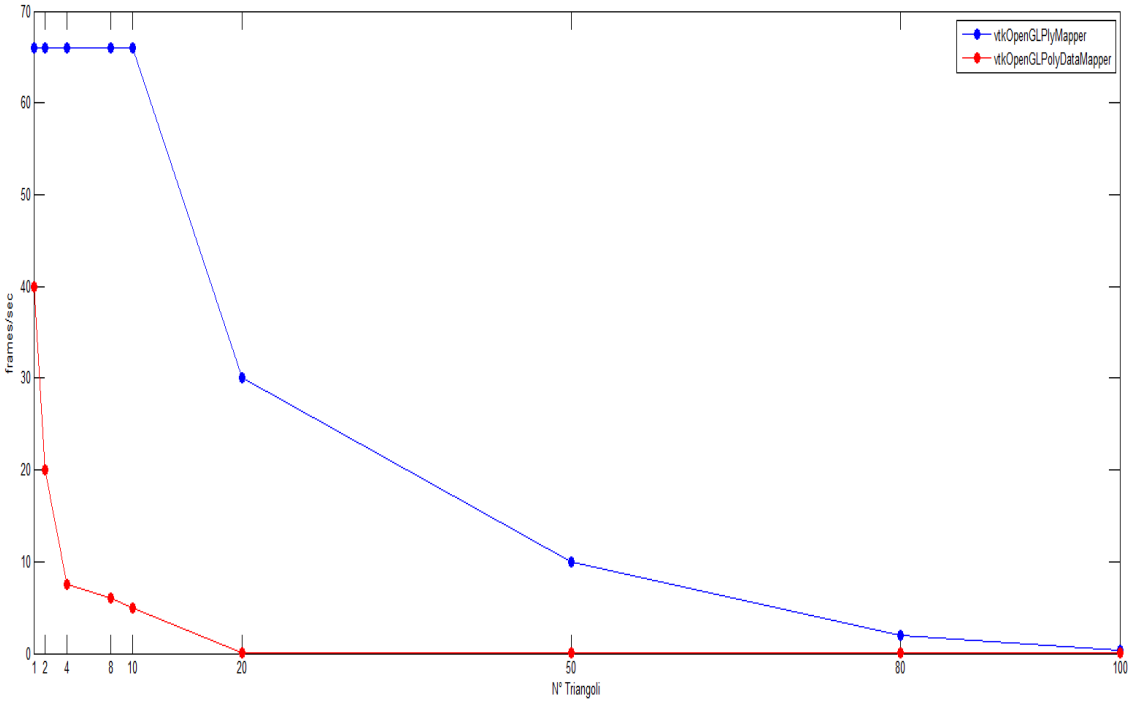


Figura 1: Confronto dei valori di frame rate ottenuti utilizzando la classe base (vtkOpenGLPolyDataMapper) e la classe derivata (vtkOpenGLPlyMapper); in ascissa è riportato il numero di triangoli del modello ply, in ordinata il numero di frame al secondo.

5. L'interfaccia grafica-wxWidgets

WxWidgets è una libreria per lo sviluppo di programmi dotati di interfaccia grafica multipiattaforma. La libreria ha la particolarità di usare i widgets grafici nativi del sistema ospite, caratteristica che la rende portabile su diverse piattaforme e sistemi operativi.

L'interfaccia realizzata per il Server e molto semplice. Essa è dotata di un menu dal quale è possibile invocare lo start/stop del programma ed impostare alcune opzioni, tra le quali la possibilità di eseguire un rendering offscreen sul server o di decidere la dimensione dell'immagine da inviare verso il client.

E' inoltre possibile specificare il numero massimo di connessioni consentite, mentre ogni volta che una connessione è accettata l'indirizzo IP del client è stampato a video.

Nella casella di testo in basso è invece riportata la lista dei file ply messi a disposizione sul sistema server.

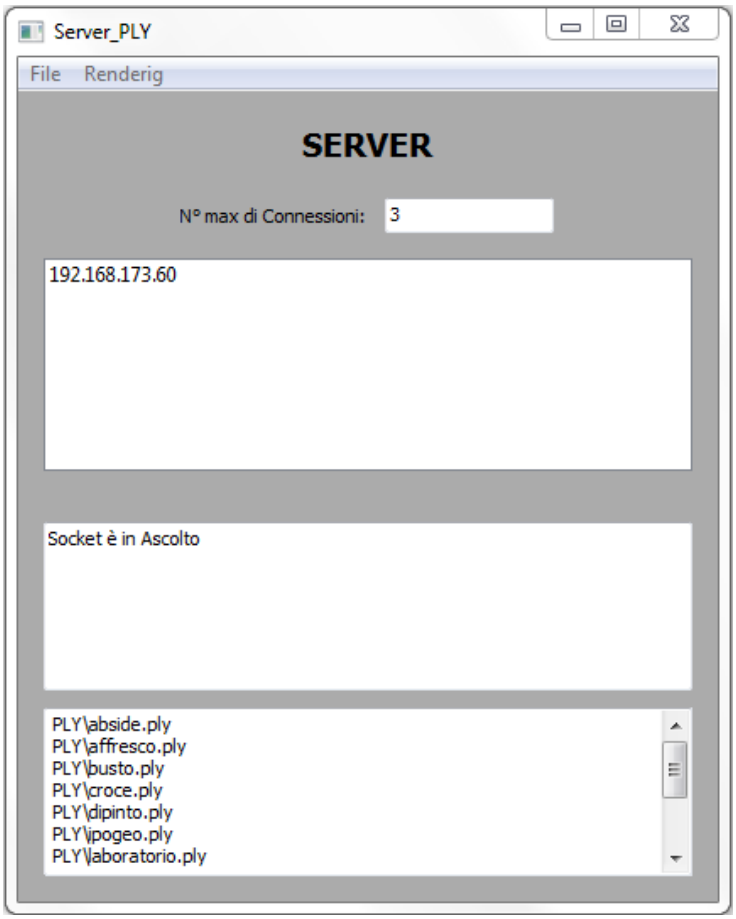


Figura 2: Interfaccia grafica del server

6. Client-Android

Per lo sviluppo del client si è deciso di utilizzare dispositivi mobili basati su sistema operativo Android. L'architettura di android è molto sofisticata e, malgrado qualche inevitabile imperfezione dovuta alla giovane età della piattaforma, rappresenta un ambiente moderno, affidabile, completo e soprattutto espandibile. Un ulteriore punto di forza di tale sistema è che la piattaforma di google costituisce un ambiente completamente open-source. Ciò ha facilitato e reso possibile una larga diffusione ed implementazione di android sui dispositivi mobili dei produttori più importanti nel panorama mondiale.

Tramite un ambiente integrato costituito da Eclipse, Android development toolkit e l'SDK di java è stata sviluppata una applicazione client basata su tre activity android.

La prima attività risolve le problematiche di connessione client/server tramite socket e protocollo TCP, la seconda presenta all'utente la lista dei file ply selezionabili, mentre la terza attività gestisce la fase di rendering e quella di interazione utente (rotazione/traslazione/zoom) con il modello 3D.

Lo scambio di messaggi tra Client e Server avviene tramite un protocollo di comunicazione basato sull'invio di brevi comandi in formato ASCII, ad esempio l'interazione con l'oggetto tridimensionale avviene traducendo gli eventi multi-touch sullo schermo tattile android in stringhe contenenti le coordinate relative al percorso tracciato dalla mano dell'utente:

mov cx-cy;cx2-cy2!"Type"

dove (cx,cy) e (cx2,cy2) sono due copie di coordinate schermo definite dall'evento di thouching; type è invece un carattere ascii che può assumere tre valori, R per rotazione, M per la traslazione e Z per lo zoom.

Il server a sua volta determina a partire da tali dati la matrice di trasformazione da applicare al modello 3D ed una volta eseguito il rendering, comprime l'immagine e la invia al client che a sua volta la decomprime e la visualizza come una bitmap nella propria interfaccia.

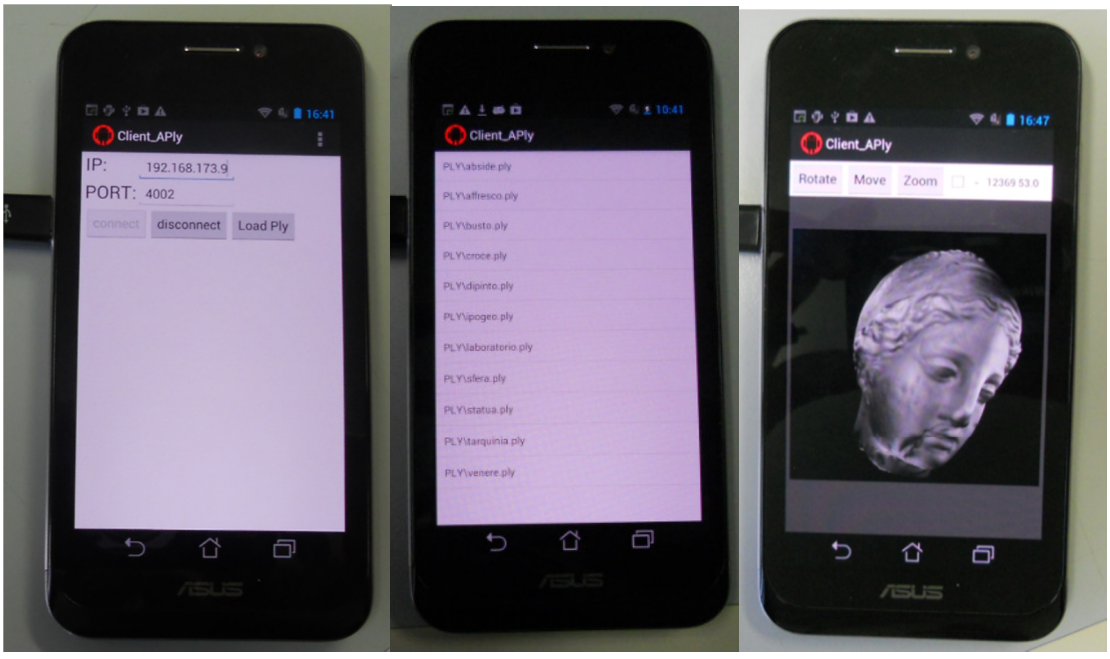


Figura 3: Le tre activity definite per il client android

7. Risultati

Le prove sono state condotte utilizzando i seguenti sistemi hardware:

- SMARTPHONE ASUS PADFONE dotato di 1Gb di RAM e sistema operativo Android 4.0

- WORKSTATION con processore Intel Core i3 S30, 16Gb di RAM e GPU Nvidia QUADRO 4000 con sistema operativo Windows 7 a 64bit.

- ACCESS POINT WIRELESS 801.11

Il client android è connesso alla rete LAN (100Mbit/sec), su cui è presente il server, attraverso un access-point WI-FI. Abbiamo condotto prove con dataset di dimensioni differenti misurando il numero di frame al secondo del processo di rendering. Il rendering OpenGL è eseguito su un device context impostato a 512x512 pixels del tipo RGBA. L'immagine risultante ha quindi una dimensione di 1Mbyte, per limitare il tempo di trasmissione, prima di essere inviata dal server essa viene compressa di un fattore 100 in formato jpeg. Nella tabella I sono riportati i valori misurati utilizzando due differenti velocità di connessione WI-FI, in particolare è indicato oltre al frame rate sul client anche il tempo di rendering del server ed il tempo necessario alla trasmissione dell'immagine jpeg. Da un'analisi dei dati si può notare come mentre il tempo di rendering è direttamente proporzionale alla dimensione del modello, il tempo di trasmissione rimane all'incirca costante all'interno di un range di valori risultando praticamente trascurabile per un numero di triangoli oltre ad una certa soglia. Per mantenere in questi casi un "interaction rate" adeguato (oltre i 20 frames/sec) il software è stato dotato di una funzionalità, attivabile dall'utente, che consente di ridurre attraverso un decimazione la dimensione del modello durante le fasi di interazione (rotazione/scaling/traslazione). La soluzione proposta consente dunque di ottenere ottimi livelli di navigabilità con elevati valori di frame rate anche in presenza di dataset di dimensioni considerevoli. L'algoritmo di compressione jpeg limita infatti l'impegno di banda per lo streaming dell'immagine introducendo ritardi minimi di trasmissione ed allo stesso tempo l'impiego dei vertex-array OpenGL permette al server di ultimare il rendering remoto dei modelli 3D in pochi msec. Con un collegamento di discreta qualità è possibile una fruizione dei dataset dei laser scanner senza dover effettuare il downloading ed utilizzare delle applicazioni di visualizzazioni in locale sul dispositivo mobile.

8. Conclusioni

In questo lavoro è descritto lo sviluppo di un'applicazione software che consente di visualizzare file in formato ply, generati da processi di scansione di laser scanner 3D, su dispositivi mobili. Onde superare limiti e vincoli tecnologici di tali dispositivi si è deciso di adottare una soluzione in cui i processi di calcolo e di rendering fossero demandati ad un'architettura remota, utilizzando il dispositivo palmare solo per la visualizzazione e l'interazione con l'utente. Tale soluzione comporta il vantaggio di poter elaborare modelli 3D di dimensioni considerevoli ed inoltre supera la necessità di dover trasmettere ed elaborare i file sorgente in locale sul dispositivo dell'utente.

Tabella I: Risultati ottenuti; la dimensione dei dataset è di circa 20MByte ogni milione di triangoli.

CONNESSIONE WI-FI 802.11n (65Mbit/sec)			
N° triangoli dataset	Frame rate Server frames/sec	Tempo di trasmissione/ricezione immagine	Frame rate Client frames/sec
$1*10^6$	66 (15 msec)	2 - 5 msec	25 (40 msec)
$2*10^6$	66 (15 msec)	2 - 5 msec	25 (39 msec)
$4*10^6$	66 (15 msec)	2 - 5 msec	25 (39 msec)
$8*10^6$	66 (15 msec)	2 - 5 msec	25 (40 msec)
$10*10^6$	66 (15 msec)	2 - 5 msec	22 (45 msec)
$20*10^6$	33 (30 msec)	2 - 5 msec	17 (60 msec)
$30*10^6$	25 (40 msec)	2 - 5 msec	14 (70 msec)
$50*10^6$	10 (100 msec)	2 – 5 msec	7 (140 msec)
$80*10^6$	2 (500 msec)	2 – 5 msec	1.6 (600 msec)
$100*10^6$	0.29 (3,44 sec)	2 - 5 msec	0.289 (3,46 sec)
CONNESSIONE WI-FI 802.11b (11Mbit/sec)			
N° triangoli dataset	Frame rate Server frames/sec	Tempo di trasmissione/ricezione immagine	Frame rate Client frames/sec
$1*10^6$	66 (15msec)	10 - 20 msec	20 (50msec)
$2*10^6$	66 (15msec)	10 - 20 msec	18 (55msec)
$4*10^6$	66 (15msec)	10 - 20 msec	18 (56msec)
$8*10^6$	66 (15msec)	10 - 20 msec	19 (52msec)
$10*10^6$	66 (15msec)	10 - 20 msec	18 (56 msec)
$20*10^6$	33 (30msec)	10 - 20 msec	14 (70 msec)
$30*10^6$	25 (40msec)	10 - 20 msec	13 (75 msec)
$50*10^6$	10 (100 msec)	10 – 20 msec	7 (140 msec)
$80*10^6$	2 (500 msec)	10 – 20 msec	1.6 (600 msec)
$100*10^6$	0.29 (3.44 sec)	10 – 20 msec	0.288 (3.47 sec)

RIFERIMENTI BIBLIOGRAFICI:

[1] Joachim Diepstraten, Martin Görke, Thomas Ertl; Remote Line Rendering for Mobile Devices. Proceedings of Computer Graphics International (CG104), pp. 454-461, 2004

[2] C.F.Chang and S.H. Ger. Enhancing 3D graphics on mobile devices by image-based rendering. In PCM '02 pages 1105-111,London,UK,2002

[3] F. Lamberti and A. Sanna, ``A streaming-based solution for remote visualization of 3D graphics on mobile devices '', IEEE Transactions on Visualization and Computer Graphics Vol. 13, No. 2, pp. 247-260, 2007

[4] F. Lamberti, C. Zunino, A. Sanna, A. Fiume, M. Maniezzo. An accelerated remote graphics architecture for PDAs. Proceeding of the eight international conference on 3D web technology, pp. 55-61, 2003

[5] W. Yoo, S. Shi, W.J.Jeon, K. Nahrstedt, R.H.Campbell; Real Time remote rendering for 3d video for mobile devices, in MM 2009, pp. 391-400, acm

[6] Visualization Toolkit An object-oriented approach to 3d graphics; Will Schroeder, Ken Martin; bill Lorensen; Kitware inc.

[7] Android Programmazione Avanzata, Emanuele di Saverio, Stefano Sanna; Edizioni FAG srl