Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# Studio tecnologico per il porting di MITO (Medical Imaging TOolkit) su dispositivi mobili

Alessio Pierluigi Placitelli
Luigi Gallo

**RT-ICAR-NA-2012-8** **Settembre 2012**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

# Studio tecnologico per il porting di MITO (Medical Imaging TOolkit) su dispositivi mobili

Alessio Pierluigi Placitelli [1]
Luigi Gallo [1]

***Rapporto Tecnico N.:***
**RT-ICAR-NA-2012-8**

***Data:***
**Settembre 2012**

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli

# Studio tecnologico per il porting di MITO (Medical Imaging TOolkit) su dispositivi mobili

Sono presentate tre strategie per il porting del software MITO su piattaforme mobile.

La prima strategia, raccomandata, prevede l'uso della libreria VES (Kitware) per eseguire i render sul dispositivo, fornendo una comoda base di sviluppo limitandosi alle piattaforme supportate.

La seconda strategia prevede l'uso delle tecnologie HTML5 e WebGL, eseguendo i render sul dispositivo, permettendo l'esecuzione del software su tutte le architetture che supportano tali standard.

La terza ed ultima strategia, prevede lo sviluppo di un thin client per ciascuna piattaforma supportata che si appoggi ad una render farm remota per l'esecuzione dei calcoli grafici.

## Sommario

# 1 Introduzione

Le immagini di tipo medicale sono comunemente usate come ausilio per la diagnosi di particolari condizioni emergenti, critiche per la salute del paziente. La digitalizzazione di questo tipo di immagini ne consente la visualizzazione da pressoché qualsiasi dispositivo di calcolo. La loro disseminazione, una volta vincolata dai limiti delle tecnologie di telecomunicazione, è stata enormemente facilitata con lo sviluppo di nuove tecnologie. Ad esempio, un Picture Archiving and Communication Systems (PACS), che memorizza immagini nel formato Digital Imaging and Communications in Medicine (DICOM), connesso mediante una rete a banda larga, permette la distribuzione di immagini mediche a piena risoluzione ovunque ve ne sia la necessità, ovvero nello stesso complesso dove risiede il server PACS o in ospedali e studi geograficamente distanti.

I continui miglioramenti tecnologici dei dispositivi mobili (smartphone e tablet), la loro maneggevolezza e il continuo aumento delle capacità hardware per l'elaborazione delle immagini rendono possibile l'uso di tali piattaforme in contesti clinici per la diagnosi preliminare e la consultazione in situazioni emergenti.

Questo rapporto presenta tre possibili strategie per il porting del software open-source MITO su dispositivi mobile. Come illustrato nel paragrafo che segue, sarà preso in considerazione per la conversione solo un sottoinsieme delle funzionalità offerte da questo software.

Nella valutazione delle differenti strategie presentate, sono stati considerati i seguenti criteri: architetture supportate, complessità di sviluppo e mantenimento del software, tempo di sviluppo richiesto ed appetibilità da un punto di vista delle problematiche di ricerca.

Le funzionalità considerate per la conversione su piattaforma mobile sono un numero ridotto rispetto alla totalità di quelle offerte da MITO.

1) Supporto PACS e DICOM
   a) Caricamento di dataset in formato DICOM da SD, allegati email, URL
   b) Autenticazione e download locale di dataset da server PACS

2) Sistema di visualizzazione avanzato
   a) Volume Rendering su GPU
   b) Supporto CLUT
   c) Supporto Windowing

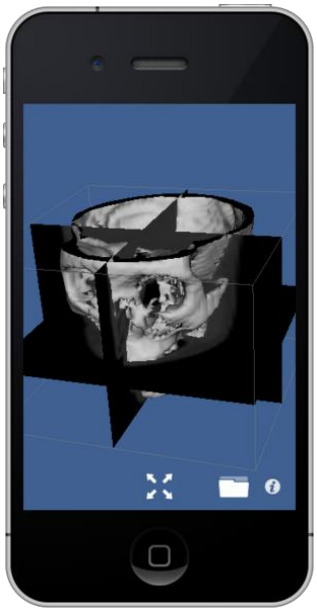
3) Sistema di interazione
   a) Funzionalità di zoom, panning e rotazione controllate da gesture su schermo sensibile al tocco
   b) Navigazione di studi e serie DICOM
   c) Misurazione di distanze ed area
   d) Funzionalità Region of Interest (ROI) per la visualizzazione a risoluzione più alta di aree di interesse circoscritte

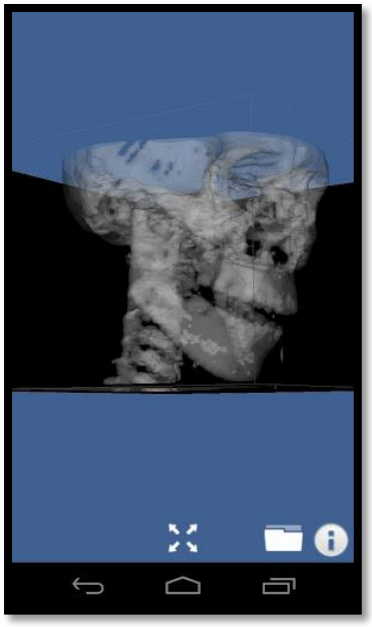
## 2 Android/iOS: Libreria nativa Kitware VES e KiwiViewer

Un software o una libreria, specificamente progettata per essere eseguito su un particolare dispositivo, architettura o ambiente software è definito nativo. Questo termine è spesso usato nel contesto di dispositivi mobili in quanto, tradizionalmente, le applicazioni venivano sviluppate per essere eseguite su dispositivi specifici. In generale, l'uso di un software nativo rispetto a software non nativo (es. applicazioni web remote e locali) offre alcuni benefici tra cui una maggiore velocità e la possibilità di accedere molto più facilmente alla risorse hardware disponibili.

Il software MITO utilizza, come base per la maggior parte delle sue funzionalità grafiche e di manipolazione, la libreria VTK (Visualization ToolKit) [1] sviluppata da Kitware. Tale libreria, purtroppo, non è progettata per l'esecuzione su architetture mobile e non supporta lo standard OpenGL ES [2], ovvero la controparte per sistemi embedded delle note OpenGL per ambienti desktop. Per colmare questa mancanza è stata sviluppata e rilasciata la libreria VES (VTK OpenGL ES Rendering Toolkit) [3], con supporto nativo per sistemi operativi iOS e Android.

Parte delle funzionalità richieste per il porting di MITO sono fornite dalla libreria VES tramite l'applicazione di visualizzazione grafica open-source KiwiViewer fornita a corredo della libreria.





L'applicazione KiwiViewer consente di collegarsi al software ParaViewWeb[1], permettendo all'utente di unirsi ad una sessione online e scaricare localmente i dati di una simulazione per la visualizzazione.

[1] Non presente nella documentazione del software ma nella descrizione di KiwiViewer nel market di Android

**Vantaggi:** supporto Android e iOS, sviluppata dalle compagnie Kitware e Willow Garage[2], integrazione con VTK, ottimizzazione per dispositivi mobile, qualità della libreria, libreria e Kiwiviewer open-source, licenza d'uso non restrittiva Apache License Version 2.0 e BSD License (possibilità d'uso in applicazioni senza il rilascio di sorgente), KiwiViewer può essere usato come thin client per ParaViewWeb.

**Svantaggi:** mancato supporto Windows 8 e Windows Phone 7, Kiwiviewer non è un singolo software cross-platform ma due software distinti (uno per Android e uno per iOS) con linee di sviluppo diverse (implica il doppio dello sforzo per la manutenzione e gli aggiornamenti), problemi nella gestione di grandi dataset.

**Stima tempi:** 200 giorni/uomo (caratteristiche 1 (a), 2 (a, c), 3 (a, b)). 150 giorni/uomo per le caratteristiche rimanenti per ciascuna piattaforma.

[2] Willow Garage cura lo sviluppo della libreria PCL (Point Cloud Library) per l'elaborazione di nuvole di punti 3D.

# 3 Sistema cross-platform basato su HTML5 e WebGL

WebGL è uno standard web cross-platform royalty-free per la grafica 3D basato su OpenGL ES 2.0, supportato da Google, Apple, Opera e Mozilla Foundation. Le funzionalità previste da tale standard sono esposte attraverso l'elemento Canvas definito nella nuova revisione del linguaggio HTML5. WebGL permette quindi di utilizzare, in qualsiasi browser che supporti lo standard, funzionalità 3D. Tra le caratteristiche di WebGL, la più interessante è sicuramente quella di essere uno standard shader-based: è quindi possibile scrivere dei piccoli software in linguaggio simil-GLSL, detti shader, da elaborare su GPU. I software scritti sono automaticamente su dispositivi mobile che utilizzano browser compatibili con HTML5 e WebGL.

Tale aspetti rendono l'uso delle tecnologie illustrate, ovvero HTML5 e WebGL, un'interessante alternativa al porting nativo di MITO su piattaforma mobile. La libreria The X Toolkit: WebGL™ for Scientific Visualization (XTK) [4], supportata da Boston's Children Hospital e Harvard Medical School, utilizza le tecnologie HTML5 e WebGL e fornisce funzionalità comparabili a quelle fornite dalla libreria VTK (volume rendering, thresholding e slicing), rendendola un'ottima candidata per l'uso in un eventuale porting di MITO con tecnologie Web.


**Vantaggi:** supporto per tutte le piattaforme conformi agli standard HTML5 e WebGL, supportata da Boston's Children Hospital e Harvard Medical School, open-source, licenza d'uso non restrittiva MIT License (possibilità d'uso in applicazioni senza il rilascio di sorgente), sviluppo di una sola applicazione.

**Svantaggi:** lo standard WebGL non è ancora universalmente supportato [5], specialmente su dispositivi mobili. Nel 2013 è previsto un aumento considerevole del supporto browser a questo standard. Performance vincolate alle prestazioni di Javascript, immaturità dello standard (Marzo 2011), problemi nella gestione di grandi dataset.

**Stima tempi:** 300 giorni/uomo (caratteristiche 1 (a), 2 (a, c), 3 (a, b)). 150 giorni/uomo per le caratteristiche rimanenti.

# 4 Thin client per rendering remoto

La visualizzazione di grandi quantità di dati su dispositivi mobili costituisce un problema di difficile risoluzione. Nonostante le capacità delle moderne GPU mobile siano in costante aumento, le loro performance continuano ad essere poco prevedibili per via dei differenti presupposti di progettazione rispetto alla controparte desktop. In ambito mobile, a differenza dell'ambito desktop, le performance di visualizzazione sono secondarie rispetto al risparmio energetico.

Una strategia che ha guadagnato nuovamente attenzione in letteratura è quella che prevede l'esecuzione dei calcoli grafici su server farm remote (render farm) e lo streaming dei risultati sui dispositivi mobili interessati. Esistono numerosi prodotti e tecnologie per l'implementazione di questo tipo di strategie, tra cui il software ParaViewWeb sviluppato da Kitware.

In questo scenario alcuni parametri di qualità della rete quali throughput e latenza diventano fondamentali nel garantire una piacevole esperienza utente. Infatti, al tempo di elaborazione della visualizzazione nella render farm remota si aggiunge il tempo di codifica dei contenuti e streaming verso il dispositivo mobile.

Un ulteriore problema è costituito dall'accesso ai dati da visualizzare: dati localizzati sul dispositivo mobile dovranno essere inviati verso la render farm per essere elaborati e visualizzati localmente.

**Vantaggi:** supporto per dataset di grandi dimensioni, disponibilità thin client mobile (KiwiViewer), licenza d'uso non restrittiva (possibilità d'uso in applicazioni senza il rilascio di sorgente).

**Svantaggi:** necessita di render farm, la latenza della rete vincola la riuscita dell'interazione utente, la banda disponibile nella rete limita il numero di client utilizzabile in parallelo, difficoltà nella visualizzazione di contenuti locali al terminale.

**Stima tempi:** n.d.

# 5 Conclusioni

Grazie alla possibilità di utilizzare il client KiwiViewer, fornito assieme alla libreria nativa VES, anche come thin client, la soluzione che prevede l'uso di client nativi è quella più economicamente appetibile. Lo svantaggio principale è quello di dover gestire, per ciascuna piattaforma supportata, un differente albero di sorgenti e quindi, sostanzialmente, differenti applicazioni.

La strategia di porting che prevede l'uso di tecnologie web, seppure tecnicamente appetibile per via del potenziale supporto multi piattaforma, non è quella raccomandata. Lo scarso supporto dei browser mobile per questi nuovi standard potrebbe prevenire l'uso di MITO mobile, nonostante nel 2013 sia previsto un incremento del supporto di WebGL da parte dei terminali mobili.

Sviluppare un thin client mobile per la visualizzazione di rendering effettuati su una render farm remota è una soluzione decisamente svantaggiosa rispetto alle strategie precedentemente illustrate, per i motivi descritti nel capitolo precedente. Un discorso diverso può essere fatto nei confronti di soluzioni ibride, ovvero applicazioni native/web che prevedano anche il supporto per rendering remoto: è questa la strada suggerita, vista la disponibilità di tecnologie off-the-shelf per la sua realizzazione.

## 6 Scheda di sintesi

| | Vantaggi | Svantaggi |
|---|---|---|
| *Applicazione Nativa* | • Libreria VES (VTK mobile)<br>• KiwiViewer per Android e iOS (open-source)<br>• Licenza non restrittiva<br>• KiwiViewer funge da thin client per ParaViewWeb | • KiwiViewer per iOS è un software diverso dalla versione Android<br>• Costi di manutenzione<br>• Doppi costi di sviluppo<br>• Mancato supporto a Windows 8 ARM per KiwiViewer e VES |
| *Applicazione HTML5+WebGL* | • Multi piattaforma<br>• Disponibilità XTK (simil VTK per WebGL)<br>• Singolo albero sorgente per ogni piattaforma<br>• Prevista, nel 2013, crescita del supporto | • Ad oggi, scarso supporto a WebGL su dispositivi mobili<br>• Performance limitate dall'uso di javascript |
| *Thin Client con Render Farm remota* | • Disponibilità del software thin client (KiwiViewer) e ParaViewWeb<br>• Gestione dataset di grandi dimensioni | • Difficoltà visualizzazione contenuti locali<br>• Costo render farm<br>• Performance limitate dalla latenza di rete e dalla banda |

# 7 Bibliografia

[1] Kitware, «VTK.org» [Online].
Available at: http://www.vtk.org/. [Consultato il 7 Settembre 2012].

[2] Khronos Group, «OpenGL ES» [Online].
Available at: http://www.khronos.org/opengles/2_X/. [Consultato il 7 Settembre 2012].

[3] Kitware, «VES» [Online].
Available at: http://www.vtk.org/Wiki/VES. [Consultato il 7 Settembre 2012].

[4] «The X Toolkit: WebGL™ for Scientific Visualization» [Online].
Available at: https://github.com/xtk/X. [Consultato il 9 Settembre 2012].

[5] «WebGL Stats» [Online].
Available at: http://webglstats.com/. [Consultato il 9 Settembre 2012].