INTERNAL REPORT

Studio di un software di ricostruzione 3D del corpo umano

Federico Spiga, Andrea Saba, Tonino Pisanu,
Mauro Pili, Pierluigi Ortu

Report N. 57,
released: 04 08 2016

Revisore: Raimondo Concu

OAC Osservatorio Astronomico di Cagliari

Sommario

1. Introduzione

Le ricerche in ambito metrologico per caratterizzare lo specchio primario del radiotelescopio e l'allineamento delle sue ottiche hanno portato allo studio di diversi sistemi di misura non a contatto. Fra questi, quelli in luce strutturata possono essere utilizzati in diversi settori della metrologia fra cui uno molto attuale e interessante è quello della caratterizzazione antropomorfica.

Il progetto prevede lo sviluppo di un sistema di scansione 3D non a contatto del corpo umano per la caratterizzazione di alcuni parametri antropometrici quali il Volume e la Superficie.

Il sistema risulta composto da: una piattaforma rotante sopra la quale viene fatto posizionare il soggetto da scansionare, un sensore di misura in luce strutturata (Kinect ONE della Microsoft), un PC contenente il software di acquisizione dei dati e quello per l'elaborazione della nuvola di punti acquisita.

Il progetto prevede inoltre, lo sviluppo del software di acquisizione e controllo del sensore con interfaccia grafica, integrato con quello di pulizia della nuvola, generazione della superficie e misura dei parametri antropometrici.

2. Introduzione alla grafica 3D

La computer grafica, CG, è la disciplina che si occupa della generazione e della manipolazione di immagini tramite il computer. Studia le tecniche e gli algoritmi per la visualizzazione di dati numerici prodotti da una macchina.

La computer grafica 3D, o modellazione 3D, è un ramo della computer grafica che tratta il processo di sviluppo di una rappresentazione matematica di una superfice a tre dimensioni di un oggetto, effettuata tramite un software specializzato. Il risultato di questo processo è chiamato modello 3D.

Un metodo di rappresentazione dei modelli 3D, utilizzato nella computer grafica 3D, sono le mesh poligonali. Una mesh poligonale, chiamata anche maglia poligonale, è una collezione di vertici, spigoli e facce che definiscono la forma di un oggetto poliedrico. Le facce sono generalmente composte da triangoli, quadrilateri o da semplici poligoni convessi per agevolare il rendering. Le mesh poligonali sono quindi delle primitive grafiche composte da strisce di triangoli o maglie di quadrilateri che permettono di visualizzare delle forme modellate in modo molto efficiente. L'unità di base delle mesh è il voxel, ossia il pixel di volume: è un elemento di volume che rappresenta un valore di intensità di segnale o di colore in un spazio tridimensionale, analogamente al pixel che rappresenta un dato in un'immagine bidimensionale.

Le mesh poligonali possono essere costruite utilizzando particolari algoritmi di triangolazione, come ad esempio l'algoritmo di triangolazione Delaunay, partendo dalle nuvole di punti. Una nuvola di punti è un insieme di punti caratterizzati dalla loro posizione in un sistema di coordinate e da eventuali valori di intensità, come il colore, ad esso associati. Vengono utilizzate per la rappresentazione di strutture tridimensionali quali oggetti e superfici in rilievo. Sono il risultato dell'utilizzo di scanner 3D e di sensori 3D.

3. Ricostruzione del mondo reale

3.1 Metodi di acquisizione 3D

Esistono molte tecniche per acquisire dei modelli 3D della realtà e che trovano applicazione nei più svariati campi. Verranno descritti i metodi più utilizzati per ricostruire un corpo 3D. Possiamo dividere i metodi in due categorie: rilevatori a contatto e rilevatori non a contatto. Tra i metodi utilizzati troviamo nei rilevatori a contatto i tastatori, mentre nei rilevatori non a contatto la fotogrammetria e la luce strutturata. Tutti questi metodi di acquisizione 3D permettono di avere come risultato una nuvola di punti che verrà utilizzata per la creazione di una mesh poligonale.

Tastatori

Tecnica basata sull'utilizzo dei tastatori, strumenti che permettono di misurare le irregolarità di una superficie. Un'evoluzione dei tastatori è la scansione di sonda. La scansione rappresenta un modo rapido per acquisire dati relativi a forma e profilo di componenti prismatici o complessi. A differenza dei tastatori a contatto, che acquisiscono punti singoli, i sistemi di scansione raccolgono un gran numero di punti sulla superficie e offrono quindi un'immagine più precisa e completa della forma e della sagoma del pezzo. La microscopia a scansione di sonda è un ramo della microscopia che forma le immagini di superfici usando una sonda fisica che esegue la scansione del campione. Un'immagine della superficie è ottenuta meccanicamente spostando la sonda in una griglia di linee, raster di scansione del campione, che muovendosi registra l'interazione sonda-superficie in funzione della posizione.

Fotogrammetria

È di rilievo la stereofotogrammetria che utilizza la stereoscopia artificiale. La stereoscopia artificiale è la riproduzione fotografica stereoscopica della realtà, creata utilizzando due punti di ripresa aventi distanza inter pupillare diversa da quella umana. La stereofotogrammetria prevede la stima delle coordinate tridimensionali di punti su un oggetto impiegando misurazioni effettuate in due immagini fotografiche scattate da due posizioni differenti, le due immagini vengono chiamate fotogrammi stereometrici. Si è quindi in grado di rilevare forma, posizione e dimensioni dell'oggetto ripreso. È inoltre possibile associare ad ogni punto il colore corrispondente.

Luce strutturata

È la tecnica che prevede di proiettare un pattern conosciuto (solitamente una griglia o delle strisce orizzontali) sulla scena. Il modo in cui questo pattern viene deformato incontrando le superfici permette al sistema di visione di calcolare la profondità e trarre informazioni sulla superficie degli oggetti. È utilizzata dagli scanner 3D a luce strutturata, tra cui troviamo il dispositivo Kinect di Microsoft. La luce strutturata utilizzata non deve interferire con gli altri dispositivi utilizzati per la visione, per questo vengono utilizzati gli infrarossi, oppure vengono proiettati due pattern esattamente opposti ad alte frequenze.

3.2 Come ricostruire una immagine 3D nel computer

Una volta effettuata l'acquisizione della nuvola dei punti si pone il problema di come tenerla in memoria e quindi salvarla nel computer.

Memorizzare e mostrare un'immagine 2D nel computer è un'operazione semplice, infatti basta utilizzare una matrice che abbia dimensione X e Y pari al numero di pixel e salvare per ciascuna coordinata le informazioni di colore del pixel. Salvare un'immagine a tre dimensioni pone il problema di dover memorizzare informazioni ulteriori alle semplici colorazioni, perciò sono state studiate delle strutture dati per il salvataggio delle mesh poligonali. Alcuni esempi sono:

- Face-vertex mesh: una semplice lista di vertici e un insieme di facce che puntano ai vertici che formano il poligono.
- Winged-edge mesh;
- Half-edge mesh;
- Quad-edge mesh;
- Corner-tables;
- Vertex-vertex mesh: rappresenta solo i vertici, che puntano ad altri vertici. Sia l'edge e le informazioni sulla faccia sono implicite nella rappresentazione. Tuttavia, la semplicità della rappresentazione consente di svolgere con maggiore efficienza diverse elaborazioni.

4. Il dispositivo Kinect

4.1 Cos'è

Il dispositivo scelto ed utilizzato per effettuare la scansione è la Kinect One o Kinect v2, rilasciata nel 2013 da Microsoft. Questo dispositivo, essendo l'evoluzione del primo modello di Kinect prodotto nel 2009, presenta dei notevoli miglioramenti hardware. Conosciuta anche con il nome di sviluppo Project Natal, la Kinect è un dispositivo di riconoscimento del movimento che permette agli utenti di controllare ed interagire con un computer usando gesti e comandi vocali. Sviluppata da Microsoft per l'intrattenimento videoludico, la Kinect ha poi rivelato il suo utilizzo in molti altri campi.
I componenti fondamentali che lavorano assieme nel sensore della Kinect sono:

- Camera RGB, aiuta il riconoscimento facciale e l'acquisizione dei colori;
- Sensore di profondità, composto da tre emettitori infrarossi (IR Emitters) e un sensore CMOS che permette di vedere l'ambiente in 3D;
- Microfoni multipli: una serie di quattro microfoni possono tracciare e isolare le voci dei giocatori dal rumore presente nella stanza.

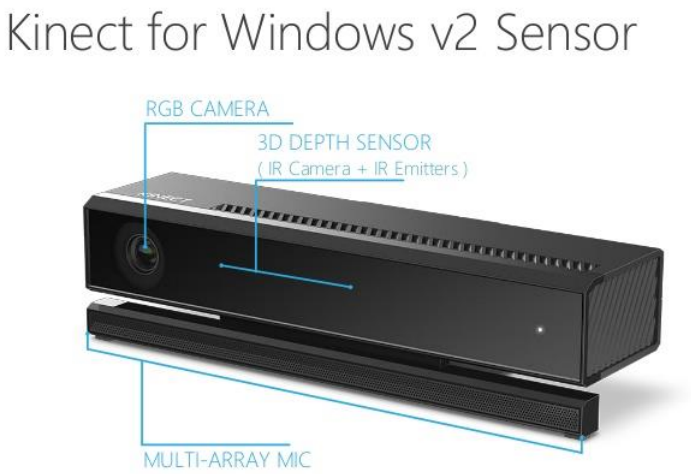

Figura 1: Kinect One

La Kinect One è in grado di rilevare il battito cardiaco, le espressioni facciali e il peso sugli arti, insieme a molti altri dati biometrici. Inoltre è in grado di monitorare individualmente le dita, le mani e le braccia e di riconoscerne l'estensione e la contrazione.

4.2 Come ricostruisce l'ambiente

Il dispositivo utilizza la tecnica della luce strutturata. Il primo passaggio effettuato dalla Kinect consiste nella creazione di un'immagine di profondità. Per crearla, è necessario avere la mappa di profondità, che tipicamente viene costruita con il processo di stereo visione. La Kinect però non adopera questo metodo,

ma utilizza la camera infrarossi e gli emettitori infrarossi di cui è dotata. Gli emettitori sparano una specifica griglia, o meglio pattern, di punti sull'ambiente e la camera cattura la luce riflessa dalla superficie colpita da ogni punto. A questo punto il processo di ricostruzione è un'operazione matematica ben definita: la disposizione dei punti del pattern è conosciuta dal software della Kinect, che al suo interno ha memorizzato come il pattern dovrebbe essere visto dal sensore IR se fosse proiettato su una superficie posta ad una distanza ben definita e perfettamente parallela al piano della camera di profondità. Per ciascun punto proiettato, si può calcolare la distanza dal sensore triangolando la sua posizione attuale, espressa in pixel, con quella che avrebbe assunto nel pattern di riferimento, in modo analogo al processo utilizzato nella stereo visione per il calcolo di una matrice di disparità. Il risultato dell'elaborazione del pattern proiettato è un'immagine di profondità grezza, ossia i valori non sono espressi in alcuna unità di misura, perciò viene effettuata una procedura di conversione in metri. La fase successiva è l'integrazione dei dati di profondità catturati dal sensore in una singola rappresentazione volumetrica dello spazio attorno alla camera. Questa integrazione dei dati di profondità viene effettuata per ciascun fotogramma, continuamente, con una media di esecuzione per ridurre il rumore e gestendo dei cambiamenti dinamici nella scena, come la rimozione o l'aggiunta di piccoli oggetti. Il dispositivo è stato progettato per rimanere in una posizione fissa, ma può anche essere mosso per vedere le superfici da più punti di vista e riempire i buchi dei dati di profondità, ad esempio la Kinect può essere ruotata attorno ad un oggetto. Il sensore di posizione può effettuare una procedura di raycasting sul volume di ricostruzione al fine di ombreggiare la nuvola di punti ottenuta, per rendere visibile un'immagine del volume di ricostruzione 3D.

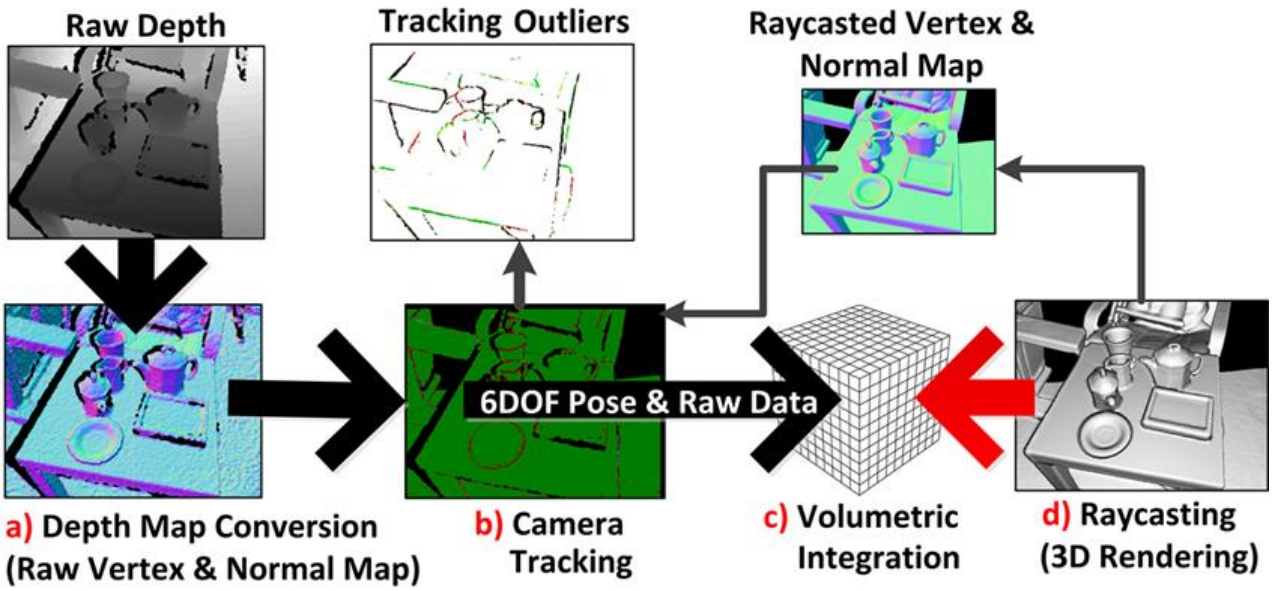

Figura 2: Flusso dei dati prodotti dalla Kinect

5. Ricostruire un corpo 3D

La ricostruzione prevede come obiettivo quello di ottenere una scansione completa e dettagliata della superficie del corpo del soggetto scannerizzato. Esistono più approcci per affrontare questa operazione, che dipendono anche dal dispositivo scelto per la scansione. Nel caso affrontato sono state prese in considerazione due possibilità: tenere fermo il corpo e far ruotare la Kinect attorno ad esso oppure tenere ferma la Kinect e far ruotare il corpo su se stesso.

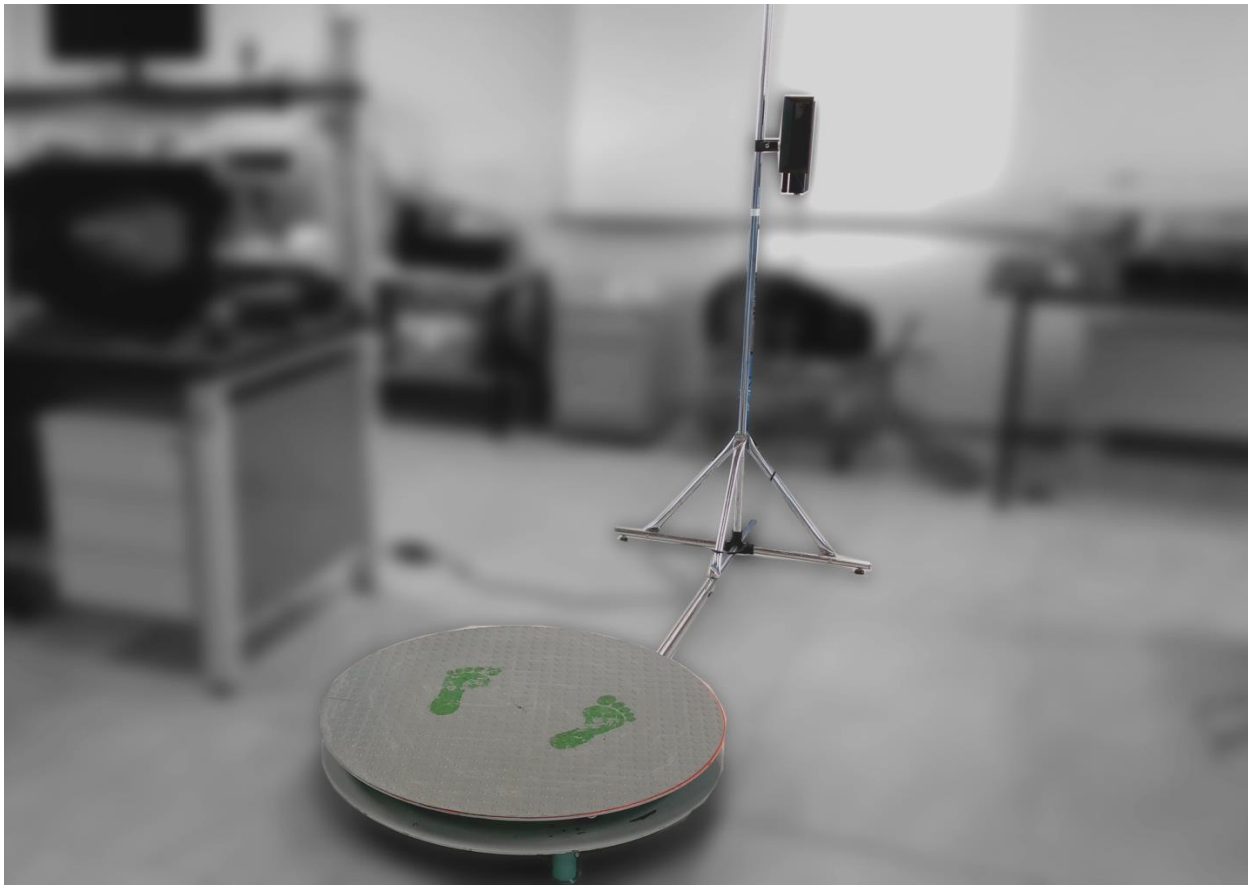

Figura 3: Struttura realizzata per la scansione

Avendo optato per la seconda scelta, è stata realizzata una struttura in grado di tenere la Kinect ferma nella stessa posizione e alla stessa distanza dal corpo da scannerizzare. Per far ruotare il corpo su se stesso, con una velocità costante e non troppo elevata, si è realizzata una piattaforma rotante sulla quale far salire il soggetto.

Per ottenere una migliore scansione si è deciso di utilizzare la Kinect con orientamento verticale, questo perché un corpo umano ha sempre una maggior altezza rispetto alla larghezza e in questo modo la scansione viene effettuata con una risoluzione maggiore.

Figura 4: Componente di supporto alla Kinect

In seguito a delle prove di scansione che hanno messo in rilievo la presenza di disturbi nel solido 3D acquisito, è stato progettato e realizzato tramite stampante 3D, un componente da montare sul dispositivo Kinect che diminuisce il raggio di visuale della camera e permette di avere meno rumore durante la scansione.

Infine per avere una buona scansione è necessario che il soggetto durante tutta la scansione rimanga il più fermo possibile sulla pedana, per diminuire l'incidenza di questo problema, è stato calcolato sperimentalmente il tempo minimo di rotazione della pedana che è stato stimato e impostato in 15 secondi circa.

6. SDK Microsoft

La Microsoft, in seguito all'utilizzo del dispositivo nei più svariati campi, ha lanciato una versione di sviluppo della Kinect per Windows e ne ha fornito la documentazione. Inoltre ha sviluppato un SDK che permette di interfacciarsi in modo semplice e rapido al dispositivo Kinect tramite una porta USB 3.0. Nell'SDK sono inclusi dei programmi che mostrano le potenzialità e le principali funzionalità del dispositivo. Microsoft ha anche messo a disposizione il codice di questi programmi di esempio, scritti in diversi linguaggi tra cui: C++, C#, HTML. L'SDK può essere suddiviso in due parti: una riguardante le gesture e una riguardante i comandi vocali. In questi programmi vengono mostrate funzionalità quali: la vista a colori, la vista di profondità, la vista infrarossi, il tracciamento facciale, il tracciamento di più corpi simultaneamente e la ricostruzione di un corpo umano.

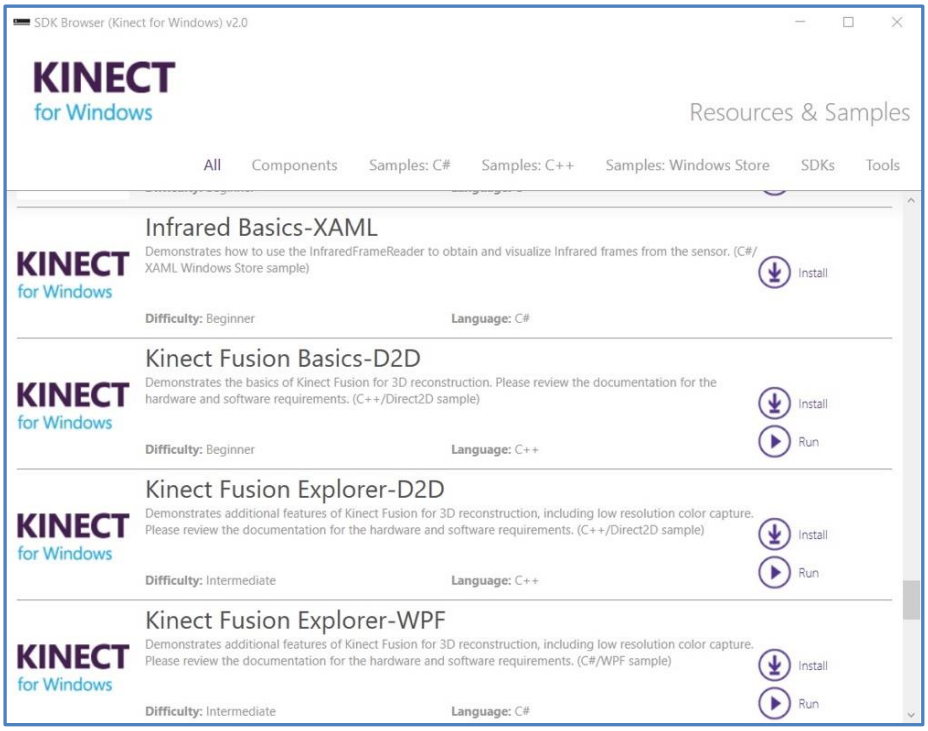

Figura 5: Interfaccia dell'SDK fornito da Microsoft

L'applicazione su cui si è focalizzata l'attenzione è il Kinect Fusion Explorer-WPF, scritto in C#, che si occupa appunto della scansione del corpo umano. La versione proposta da Microsoft è molto completa, presenta infatti un'interfaccia grafica che comprende diversi tipi di viste della camera, permette di impostare molti parametri e attivare impostazioni per la scansione. Le viste messe a disposizione sono: profondità, riconoscimento del corpo e ricostruzione della mesh. Attraverso l'uso del mouse, il programma permette di spostare la visuale in cui è presente la ricostruzione del corpo, permettendone la visione da infinite angolazioni. I parametri regolabili per la scansione sono i seguenti:

- Risoluzione per ciascuno dei tre assi X, Y, Z (lunghezza, larghezza, altezza);
- Distanza minima e massima di acquisizione dell'immagine;
- Numero di Voxel per metro;
- VolumeMaxIntegrationWeight.

Inoltre permette di utilizzare opzioni quali:

- Cattura del colore;
- Riflessione della visuale;
- Visione della scansione in uno spazio tridimensionale;
- Reset e pausa dell'acquisizione;
- Reset della visuale della camera;
- Salvataggio della mesh in più formati (stl, obj e ply).

L'ultima feature inclusa nell'applicazione è una status bar situata nella parte inferiore della finestra che mostra messaggi sull'operazione in corso o sulle problematiche che si stanno verificando.

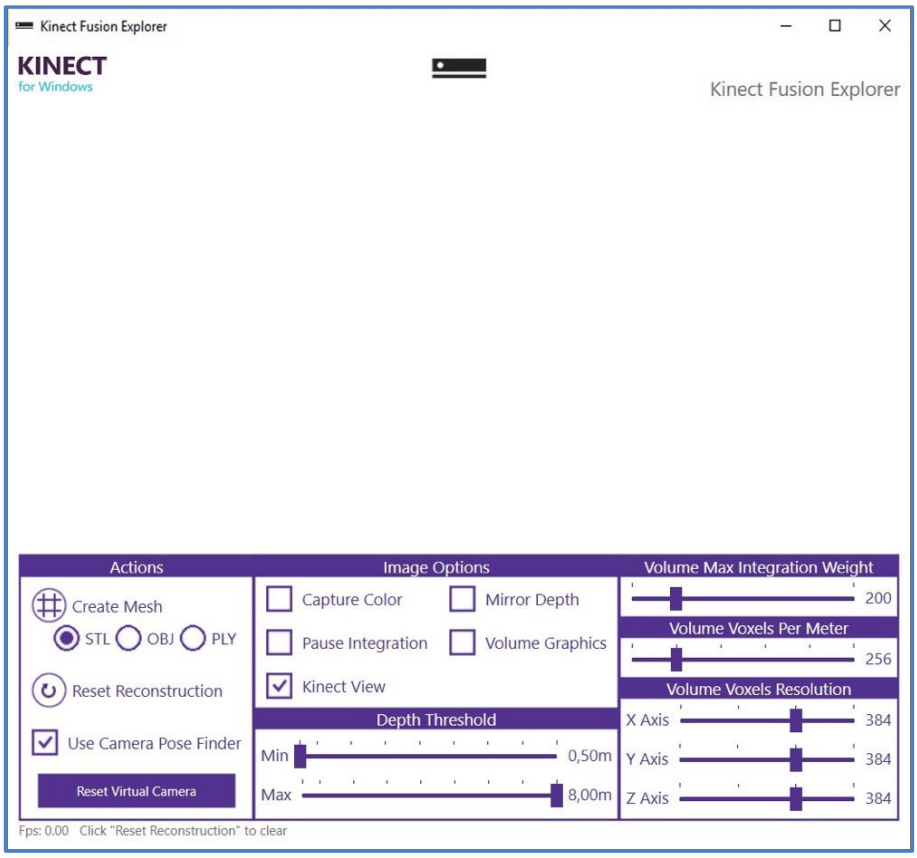

Figura 6: Interfaccia di Kinect Fusion Explorer-WPF

7. Realizzazione del software di acquisizione

7.1 Scelte implementative

Per lo scopo prefissato dal progetto, si è deciso di partire dal programma Kinect Fusion Explorer-WPF semplificandone l'interfaccia grafica e studiando la miglior configurazione dei parametri al fine di avere una buona acquisizione utilizzando la struttura creata appositamente per la Kinect. L'interfaccia grafica è stata ripulita da tutti i pulsanti, checkbox e slider presenti, limitandosi a mostrare solamente: un pulsante di avvio per la scansione, una barra di caricamento e le tre visuali, senza dare la possibilità di muovere l'inquadratura. È stata introdotta la possibilità di caricare i parametri da un file di testo molto semplice da realizzare, composto riga per riga da: nome del parametro, carattere '=', valore del parametro e nel caso in cui si voglia aggiungere un commento si inserisce il carattere '#' seguito dal commento. Nella Figura 7 si può vedere l'esempio di un file di configurazione.

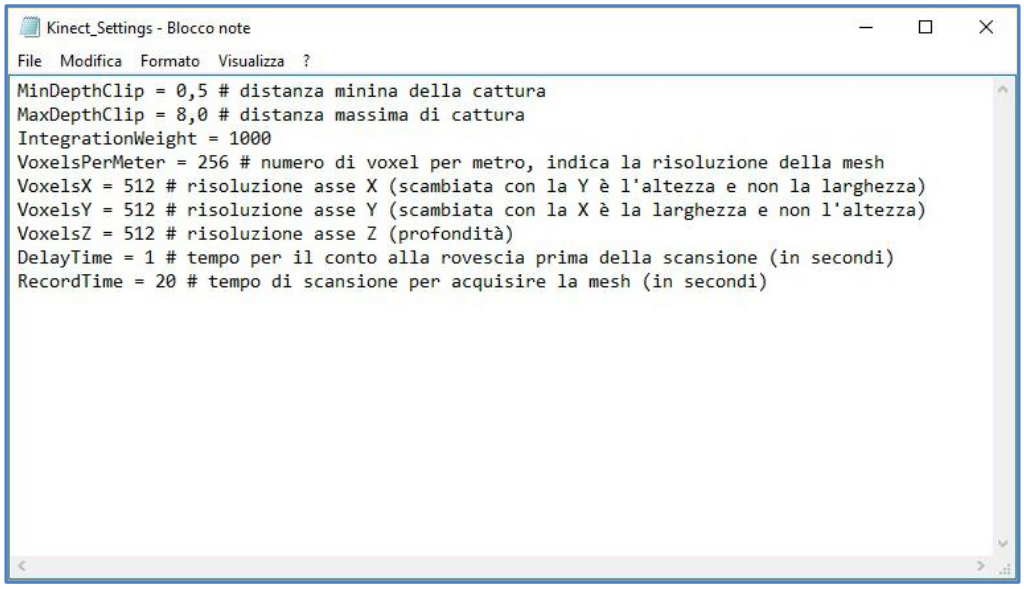

Figura 7: Esempio di file di configurazione dei parametri

Un'altra modifica implementata è quella di poter impostare due temporizzazioni, la prima viene utilizzata come countdown per permettere al soggetto di posizionarsi correttamente sulla pedana; la seconda invece viene utilizzata dal software per impostare la durata della scansione del corpo.

Si è scelto di effettuare il salvataggio della mesh solo in formato STL (STereo Lithography interface format oppure Standard Triangulation Language). È un formato di file nato per i software di stereolitografia CAD e utilizzato nella prototipazione attraverso software CAD e rappresenta un solido la cui superficie è stata discretizzata in triangoli. Il formato STL presenta dei vantaggi quali la semplicità, in quanto risulta molto facile da generare e da processare, mentre a suo sfavore presenta una geometria approssimata e la sua struttura dati può presentare la ripetizione dello stesso vertice più volte. I file in formato STL possono essere visualizzati o corretti con strumenti open source come MeshLab o commerciali. Inoltre è uno dei principali formati usati nell'ambito della stampa 3D.

Si è anche pensato ad un utilizzo multiplo del programma, ossia permettere di effettuare più scansioni senza dover chiudere e riaprire il programma. Per sviluppare questa funzionalità si è pensato di chiudere la lettura dei dati da Kinect e di scollegarla, tramite software, e di ricollegarla solo quando l'utilizzatore

volesse effettuare un'altra scansione, ma è stato riscontrato un problema software che impedisce di ricollegare la Kinect una volta che è stata scollegata. Per questo si è optato di non scollegare la Kinect, ma solamente di resettare i dati acquisiti e di mettere in pausa l'integrazione dei dati. A questo punto le due scelte possibili erano: impedire di effettuare più scansioni spegnendo il sensore Kinect dopo aver effettuato la scansione o permettere di effettuare più scansioni lasciando il sensore acceso. La scelta adottata è stata la seconda, che come compromesso implica di tenere accesa la Kinect da quando si effettua la prima scansione a quando si chiude il programma.

7.2 Studio dei parametri

Il volume di ricostruzione è composto da piccoli cubi nello spazio, che normalmente vengono chiamati Voxels. Il numero di voxels che possono essere creati dipende dalla quantità di memoria disponibile ad essere allocata per il dispositivo di ricostruzione, tipicamente si utilizzano 640x640x640 = 262144000 voxels, che possono essere creati in totale su un dispositivo con 1.5GB di memoria o superiore. L'aspect ratio del volume può essere arbitrario.

L'unità voxelsPerMetro indica la dimensione che 1 voxel rappresenta nel mondo reale, quindi avendo un volume cubico di 384x384x384 esso può rappresentare un cubo di 3m nel mondo reale se si sceglie come voxelsPerMetro un valore pari a 128vpm (384/128 = 3, in cui ogni voxel è 3m/384 = 7.8 mm^3), o un cubo di 1.5m se si utilizza un valore pari a 256vpm (384/256 = 1.5, in cui ogni voxel è 1.5m/384 = 3.9 mm^3). Questa combinazione di voxels negli assi X, Y e Z e di voxelsPerMetro permette di specificare un volume con diverse dimensioni e risoluzioni, ma è un compromesso con il numero fisso di voxels che possono essere creati, non è possibile infatti creare un ampio volume con un'alta risoluzione.

Per la scansione 3D del corpo umano è stata studiata una configurazione che permettesse di avere la più alta risoluzione possibile tenendo conto della dimensione necessaria a far inquadrare la figura umana all'interno del volume. Scegliendo come limite massimo di altezza del corpo 2m è stato necessario impostare un valore pari a 256vpm di voxelsPerMetro e una risoluzione sull'asse verticale uguale a 512 (512/256 = 2). A questo punto, come risoluzione per gli altri due assi si è scelto un valore di 256. Questa configurazione permette di costruire un volume alto 2m, largo 1m e profondo 1m. Nell'impostazione dei parametri bisogna tener conto che gli assi X e Y sono ruotati di 90° sull'asse Z poiché la Kinect è utilizzata con orientamento verticale.

I successivi parametri da configurare sono la distanza minima e massima di acquisizione della visione di profondità, se regolati in modo corretto permettono di ottenere una notevole riduzione del rumore. Avendo costruito una struttura che tiene ferma la Kinect e sapendo che la pedana rotante si trova sempre alla stessa distanza, è stato possibile calcolare i due parametri in modo preciso. Sono stati scelti i seguenti valori: per la distanza minima 1m e per la distanza massima 2m.

Gli ultimi due parametri da configurare sono le due temporizzazioni. Avendo stabilito che la pedana, utilizzata con un'alimentazione di 13V, impiega circa 14s per completare un giro intero a 360° si è scelto un tempo di scansione di default pari a 15s. Il tempo di countdown non influisce sul risultato della scansione ed è stato scelto un valore arbitrario di default pari a 5s.

7.3 Funzionamento del software

All'avvio del software verrà aperta la finestra principale in cui è presente come unica possibilità di interazione quella di premere il bottone 'Start Scan'.

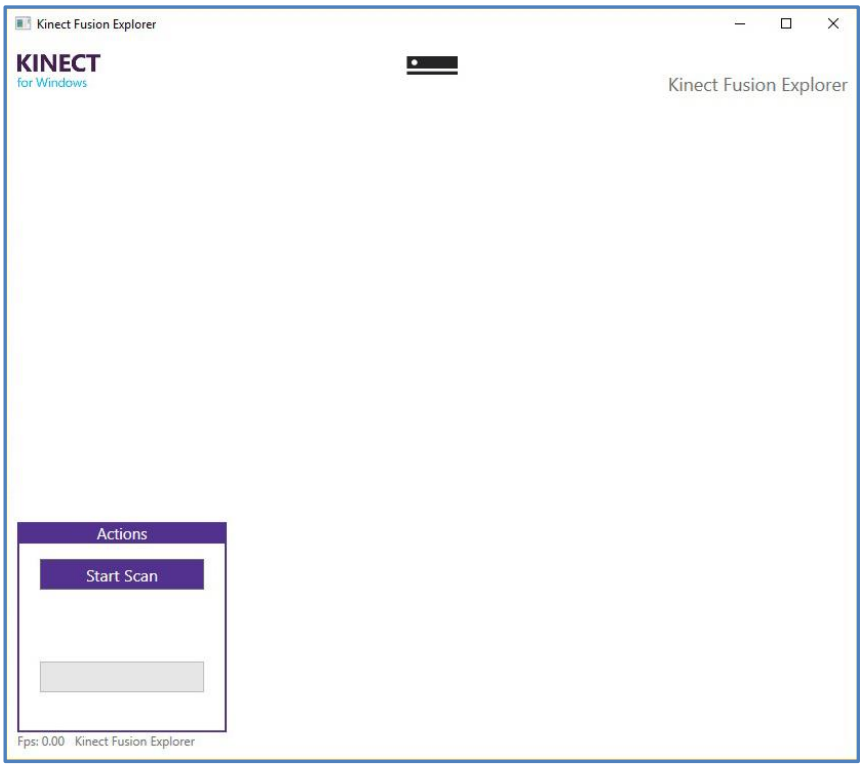

Figura 8: Schermata principale

A questo punto il bottone di inizio scansione verrà reso non cliccabile e verrà aperta una finestra di dialogo che chiede se si vogliano utilizzare i parametri di default o se si voglia usare una configurazione personalizzata. Scegliendo una configurazione personalizzata si aprirà una finestra che permetterà di scegliere un file di testo (file .txt) tra le risorse presenti nel computer.

Dal momento della selezione del file, in caso di configurazione personalizzata, o dalla selezione della configurazione di default partirà il countdown per l'inizio della scansione, che verrà visualizzato tramite la barra di caricamento. Al termine del countdown partirà la scansione e verranno mostrate le tre visuali della Kinect nella finestra principale. Anche qui sarà possibile vederne lo stato tramite la barra di caricamento.

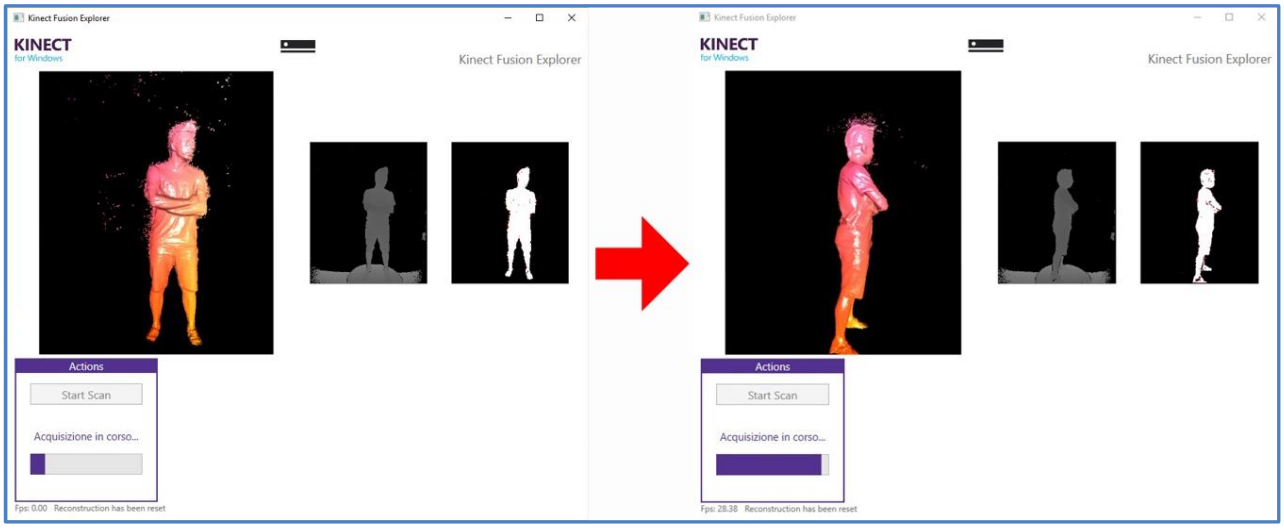

Figura 9: Schermata di funzionamento

A fine scansione verrà aperta in modo automatico la finestra per il salvataggio della mesh.

A questo punto verranno reimpostati in modo automatico i parametri ai valori di default, verrà resettata l'acquisizione della Kinect, l'integrazione dei dati verrà messa in pausa e il bottone di inizio scansione verrà reso nuovamente cliccabile. L'utilizzatore potrà decidere quindi se effettuare un'altra scansione premendo nuovamente il bottone di inizio scansione o potrà uscire dal programma chiudendo la finestra.

Figura 10: Ricostruzione eseguita correttamente

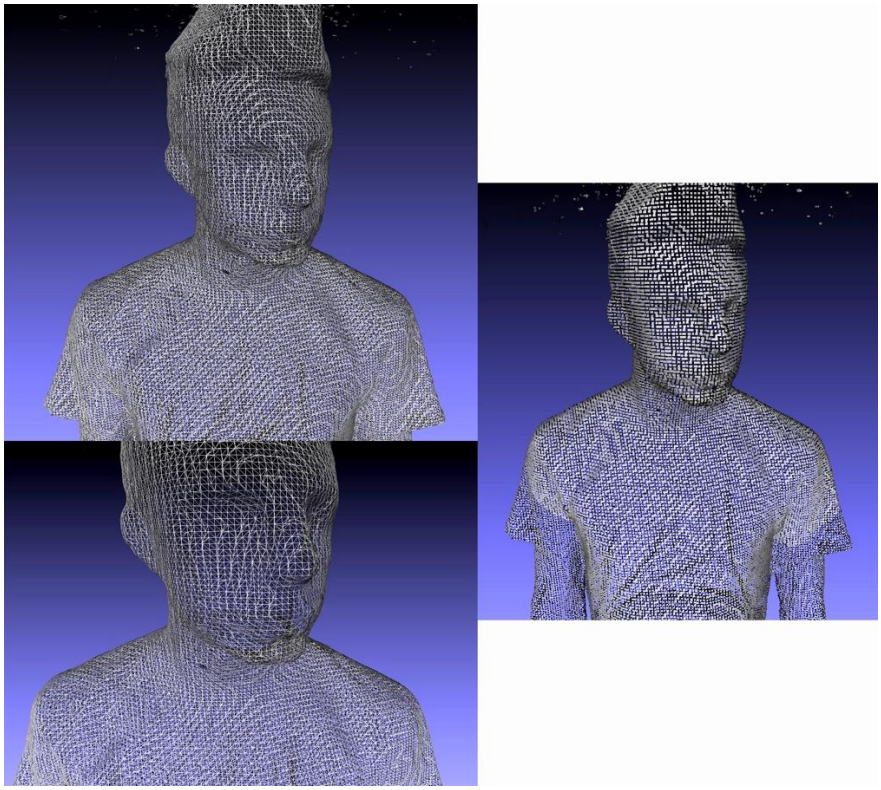

Figura 11: Dettagli mesh

7.4 Sistema, ambiente di sviluppo e linguaggi utilizzati

Per poter utilizzare la Kinect One, che si interfaccia tramite USB 3.0, è stato necessario utilizzare un PC performante e con una buona scheda grafica. La macchina utilizzata per la realizzazione del software è dotata di: processore Intel Core i7-4790 a 3,6 GHz, 8 GB di memoria DDR3, un hard disk da 1TB, una scheda video grafica Quadro K4000 – PNY, Windows 10 professional a 64 Bit.

Figura 12: IDE e linguaggi utilizzati

L'applicazione è stata sviluppata tramite l'IDE Microsoft Visual Studio. Sono stati utilizzati i linguaggi C# e XAML.

Visual Studio è un ambiente di sviluppo integrato realizzato da Microsoft, che supporta diversi tipi di linguaggio, tra i quali: C, C++, C#, F#, Visual Basic .Net e ASP .Net. Permette di creare applicazioni, siti web, applicazioni web e servizi web. È multipiattaforma, Visual Studio permette di realizzare programmi per server, workstation, pocket PC, smartphone e browser.

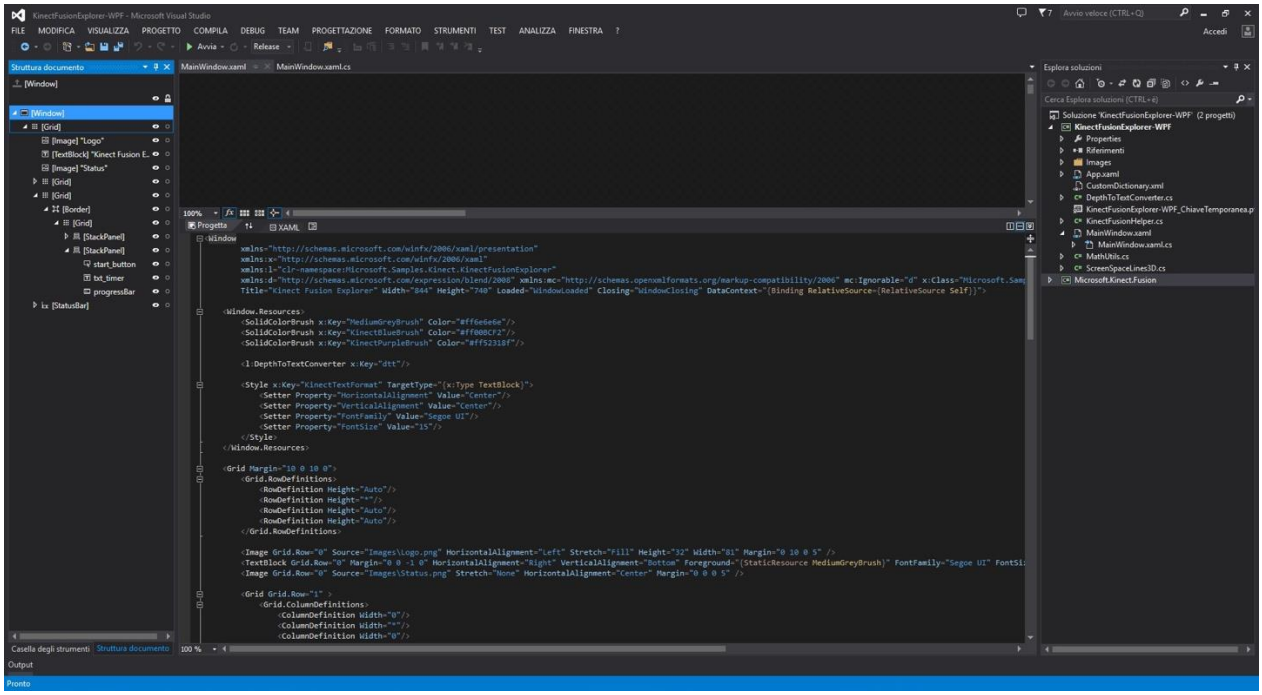

Figura 13: Schermata Microsoft Visual Studio

È stata sviluppata un'applicazione di tipo WPF che permette di sviluppare applicazioni utilizzando il markup e il code-behind. Il markup XAML viene utilizzato per l'implementazione dell'aspetto dell'applicazione, per le interfacce utente, mentre i linguaggi di programmazione code-behind vengono usati per implementarne il comportamento, ossia le risposte alle interazioni dell'utente e la gestione degli eventi. Nel caso trattato il linguaggio code-behind utilizzato è il C#.

7.5 Codice dell'applicazione

Sono di seguito riportate e commentate le funzioni che si occupano delle parti fondamentali dell'applicazione. Sono state scelte tre funzioni: la StartKinect, la Reader_MultiSourceFrameArrived e la WindowClosing.

La StartKinect ha il compito di collegare il dispositivo Kinect, di abilitare in ingresso il flusso dei dati della camera di profondità e della camera a colori e di preparare le risorse necessarie alla ricostruzione. È chiamata dopo aver premuto il bottone di inizio scansione e dopo aver scelto il settaggio dei parametri.

```
private void StartKinect()
{
  // One sensor is supported
  this.sensor = KinectSensor.GetDefault();

  if (null == this.sensor)
  {
     this.statusBarText.Text = Properties.Resources.NoReadyKinect;
     return;
  }

  // get the coordinate mapper
  this.mapper = this.sensor.CoordinateMapper;
```

```
  // open the sensor
  this.sensor.Open();

  this.reader = this.sensor.OpenMultiSourceFrameReader( FrameSourceTypes.Depth |
                                                        FrameSourceTypes.Color );

  FrameDescription depthFrameDescription =
                                    this.sensor.DepthFrameSource.FrameDescription;
  this.depthWidth = depthFrameDescription.Width;
  this.depthHeight = depthFrameDescription.Height;
  this.depthPixelCount = this.depthWidth * this.depthHeight;

  FrameDescription colorFrameDescription =
                                    this.sensor.ColorFrameSource.FrameDescription;
  this.colorWidth = colorFrameDescription.Width;
  this.colorHeight = colorFrameDescription.Height;
  this.colorPixelCount = this.colorWidth * this.colorHeight;

  this.depthVisibilityTestMapWidth = this.colorWidth / ColorDownsampleFactor;
  this.depthVisibilityTestMapHeight = this.colorHeight / ColorDownsampleFactor;
  this.depthVisibilityTestMap = new ushort[this.depthVisibilityTestMapWidth *
                                        this.depthVisibilityTestMapHeight];

  // Start worker thread for depth processing
  this.StartWorkerThread();

  // Start fps timer
  this.fpsTimer = new DispatcherTimer(DispatcherPriority.Send);
  this.fpsTimer.Interval = new TimeSpan(0, 0, FpsInterval);
  this.fpsTimer.Tick += this.FpsTimerTick;
  this.fpsTimer.Start();

  // Set last fps timestamp as now
  this.lastFPSTimestamp = DateTime.UtcNow;

  // Start status bar timer
  this.statusBarTimer = new DispatcherTimer(DispatcherPriority.Send);
  this.statusBarTimer.Interval = new TimeSpan(0, 0, StatusBarInterval);
  this.statusBarTimer.Tick += this.StatusBarTimerTick;
  this.statusBarTimer.Start();

  this.lastStatusTimestamp = DateTime.Now;

  // Add an event handler to be called whenever depth and color both have new data
  this.reader.MultiSourceFrameArrived += this.Reader_MultiSourceFrameArrived;

  // Allocate frames for Kinect Fusion now a sensor is present
  this.AllocateKinectFusionResources();

  // Set recreate reconstruction flag
  this.recreateReconstruction = true;
}
```

La Reader_MultiSourceFrameArrived viene richiamata ogni qual volta si hanno nuovi dati in ingresso dalla camera di profondità o dalla camera a colori. Avviene quindi l'integrazione dei nuovi dati raccolti con quelli già catturati in precedenza.

```
private void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
{
   bool validDepth = false;
   bool validColor = false;

   MultiSourceFrameReference frameReference = e.FrameReference;

   MultiSourceFrame multiSourceFrame = null;
   DepthFrame depthFrame = null;
   ColorFrame colorFrame = null;

   try
   {
      multiSourceFrame = frameReference.AcquireFrame();

      if (multiSourceFrame != null)
      {
         // MultiSourceFrame is IDisposable
         lock (this.rawDataLock)
         {
            ColorFrameReference colorFrameReference =
                                          multiSourceFrame.ColorFrameReference;
            DepthFrameReference depthFrameReference =
                                          multiSourceFrame.DepthFrameReference;

            colorFrame = colorFrameReference.AcquireFrame();
            depthFrame = depthFrameReference.AcquireFrame();

            if ((depthFrame != null) && (colorFrame != null))
            {
               // Save frame timestamp
               this.RelativeTime = depthFrame.RelativeTime;

               FrameDescription colorFrameDescription =
                                                colorFrame.FrameDescription;
               int colorWidth = colorFrameDescription.Width;
               int colorHeight = colorFrameDescription.Height;

               if ((colorWidth * colorHeight * sizeof(int)) ==
                                               this.colorImagePixels.Length)
               {
                  colorFrame.CopyConvertedFrameDataToArray(this.colorImagePixels,
                                                   ColorImageFormat.Bgra);

                  validColor = true;
               }

               FrameDescription depthFrameDescription =
                                                depthFrame.FrameDescription;
               int depthWidth = depthFrameDescription.Width;
               int depthHeight = depthFrameDescription.Height;

               if ((depthWidth * depthHeight) == this.depthImagePixels.Length)
               {
                  depthFrame.CopyFrameDataToArray(this.depthImagePixels);

                  validDepth = true;
```

```
                }
            }
        }
    }
}
catch (Exception)
{
    // ignore if the frame is no longer available
}
finally
{
    // MultiSourceFrame, DepthFrame, ColorFrame, BodyIndexFrame are IDispoable
    if (depthFrame != null)
    {
        depthFrame.Dispose();
        depthFrame = null;
    }

    if (colorFrame != null)
    {
        colorFrame.Dispose();
        colorFrame = null;
    }

    if (multiSourceFrame != null)
    {
        multiSourceFrame = null;
    }
}

if (validDepth)
{
    // Signal worker thread to process
    this.depthReadyEvent.Set();
}

if (validColor)
{
    // Signal worker thread to process
    this.colorReadyEvent.Set();
}
}
```

La WindowClosing viene eseguita alla chiusura della finestra e si occupa non soltanto di chiudere la finestra, prima infatti vengono avviate le procedure per fermare la lettura dei dati dalle due camere, a colori e di profondità, e successivamente viene rimosso il collegamento al dispositivo Kinect.

```
private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)
{
    // Stop timer
    if (null != this.fpsTimer)
    {
        this.fpsTimer.Stop();
        this.fpsTimer.Tick -= this.FpsTimerTick;
    }

    if (null != this.statusBarTimer)
```

```
    {
        this.statusBarTimer.Stop();
        this.statusBarTimer.Tick -= this.StatusBarTimerTick;
    }

    if (this.reader != null)
    {
        this.reader.Dispose();
        this.reader = null;
    }

    if (null != this.sensor)
    {
        this.sensor.Close();
        this.sensor = null;
    }

    // Stop worker thread
    this.StopWorkerThread();
}
```

8. Conclusioni e sviluppi futuri

Il programma funziona in modo corretto e permette di effettuare la ricostruzione in modello 3D di una figura umana in modo semplice ed intuitivo. Nella versione attuale è necessario azionare manualmente la pedana rotante tramite un alimentatore esterno, è prevista la progettazione e la realizzazione di una logica elettronica in grado di pilotare il sistema di rotazione. Infatti sarà necessario predisporre un controllore in grado di pilotare la pedana rotante che sarà attivata tramite software. Inoltre sarà possibile tramite apposita sensoristica, rilevare ulteriori informazioni dal soggetto posto in scansione come ad esempio peso, ecc. Questo software è la prima parte di una pipeline, che oltre l'acquisizione si occupa di estrarre ed analizzare i dati biometrici del soggetto scansionato. Gli altri due passi della pipeline vengono eseguiti tramite un programma di modellazione 3D che si occuperà di svolgere le operazioni di: rotazione e taglio della mesh, il taglio consiste nel ripulire la mesh dal rumore derivante dall'acquisizione. Successivamente verrà effettuata l'operazione di chiusura del solido 3D, la Kinect non riesce a completare la figura sotto i piedi e nella parte superiore della testa a causa dei molti dettagli offerti dai capelli, perciò è necessario farlo successivamente all'acquisizione. L'ultimo passo della pipeline è l'estrazione e l'analisi dei parametri biometrici, sarà data la possibilità di effettuare misurazioni su dati come volume e superfice dell'intera figura o di sezioni di essa.

Il punto di arrivo del progetto è quello di integrare i due software, acquisizione e modellazione, in un unico software che svolga tutti i compiti.

Per quanto riguarda il lato hardware, la struttura realizzata tutt'ora è troppo ingombrante e non è facilmente trasportabile, si potrebbe pensare una riprogettazione della stessa che sia più funzionale.

9. Bibliografia

[1] Scateni Riccardo, corso universitario L.M. Informatica, materiale Algoritmi e Strutture Dati 2
[2] Di Ruberto Cecilia, corso universitario L.M. Informatica, materiale Elaborazione ed Analisi di Immagini
[3] http://www.makerbot.com/
[4] https://msdn.microsoft.com/
[5] https://www.visualstudio.com/
[6] http://www.luigimaggio.altervista.org/documenti/tesinaRealtaVirtualeKinect.pdf
[7] https://it.wikipedia.org/wiki/Microsoft_Visual_Studio
[8] https://it.wikipedia.org/wiki/C_sharp
[9] https://en.wikipedia.org/wiki/Kinect
[10] https://en.wikipedia.org/wiki/Kinect_for_Xbox_One
[11] http://electronics.howstuffworks.com/microsoft-kinect2.htm
[12] http://zugara.com/how-does-the-kinect-2-compare-to-the-kinect-1
[13] http://www.directindustry.it
[14] http://www.twintec.it/3dscanner.htm
[15] https://it.wikipedia.org/wiki/Fotogramma_stereometrico
[16] https://en.wikipedia.org/wiki/Photogrammetry
[17] https://it.wikipedia.org/wiki/Computer_grafica_3D
[18] https://en.wikipedia.org/wiki/3D_modeling
[19] https://it.wikipedia.org/wiki/Ray_tracing
[20] http://epiantohobbies.jimdo.com/la-computer-grafica/breve-storia-della-computer-grafica/
[21] https://it.wikipedia.org/wiki/Nuvola_di_punti
[22] https://en.wikipedia.org/wiki/Point_cloud
[23] https://en.wikipedia.org/wiki/Structured_light
[24] https://en.wikipedia.org/wiki/3D_scanner#Structured_light
[25] https://en.wikipedia.org/wiki/Solid_modeling
[26] https://it.wikipedia.org/wiki/Mesh_poligonale
[27] https://it.wikipedia.org/wiki/STL_(formato_di_file)
[28] https://en.wikipedia.org/wiki/Polygon_mesh
[29] https://it.wikipedia.org/wiki/Triangolazione
[30] https://it.wikipedia.org/wiki/Triangolazione_di_Delaunay
[31] https://it.wikipedia.org/wiki/Marching_cubes