**INTRODUZIONE GENERICA**

La generazione del dataset sintetico avviene tramite un software, reperito sul web[[1]](#footnote-1) e adattato alle specifiche del progetto, chiamato *SyntheticDataGenerator*, operante sul motore grafico Unity 3D (versione 2019.2.0f1) sotto il controllo di script C#.

Il compito di questo software è quello di generare immagini contenenti un modello 3D dell’oggetto (o degli oggetti) della detection, e di effettuarne il labelling in un formato leggibile dallo strumento che si prenderà carico del training del detector. Nel caso rappresentato da questo lavoro di tesi, l’oggetto della detection è una mano e il software che si occupa del training è TensorFlow (versione 1.15), che allenerà una sua API per l’object detection.

Il connubio tra immagini e descrizioni delle stesse è ciò che andrà a comporre concretamente il dataset sintetico generato con il software in questione.

**STRUTTURA PROGETTO**

[immagine albero cartelle]

Per descrivere i file che compongono il progetto, verrà effettuato un focus sulla cartella *Assets*[[2]](#footnote-2) e, in particolare, sulle sue subdirectories, contenenti tutto ciò che serve per rendere il progetto funzionante.

Ogni elemento contenuto nella cartella *Assets* implica la presenza di un file omonimo con estensione *.meta*, creato da Unity al momento dell’importazione nel workspace dell’elemento in questione e del quale non risulta di alcun interesse, ai fini del lavoro di tesi, il contenuto.

Come emergerà nel paragrafo *Assets/Scenes*, cartella che verrà approfondita per ultima in quanto dipendente da tutte le altre sottocartelle, la presenza di un file all’interno di *Assets*  o di una delle sue subdirectories non implica il suo utilizzo durante il processo di generazione dati: vi è infatti differenza tra importare un file nel workspace Unity, inserendolo in *Assets*, e importarlo nella scena (insieme di oggetti 3D, governati da script, posti nello spazio tridimensionale) che il motore grafico sfrutterà per generare il dataset.  
È altresì importante far notare che non vi può essere importazione nella scena di un file senza che questo sia stato già importato nel workspace.

***Assets/3DModels***

[immagine contenuto 3DModels]

La cartella *3DModels*, come intuibile dal nome, è la cartella che ospita i modelli 3D che possono essere utilizzati per la generazione del dataset.

La versione di Unity utilizzata durante il lavoro di tesi supporta modelli 3D in formato non proprietario *.fbx, .dae, .3ds, .dxf* e *.obj*. Sono supportati anche formati proprietari, previa conversione in *.fbx* presa in carico da Unity stesso.

Per la generazione di un dataset adatto al training dell’hand detector, è stato utilizzato un modello 3D di mano, reperito sul web[[3]](#footnote-3), in formato *.obj* e contenuto nella cartella *hand*.  
Il file *hand.obj* all’interno dell’omonima directory, contenente la riproduzione mediante una maglia poligonale (polygon mesh) di una mano, va integrato con il file *hand\_mapNew.mat*, dipendente da *hand\_mapNew.jpg*, che fornisce copertura alla mesh, aggiungendole la pelle altrimenti mancante e dettagli come unghie e rughe.

***Assets/GUI***

All’interno della cartella *GUI* (abbreviazione di Graphical User Interface) sono contenute, nell’unico file presente, tutte le informazioni relative all’aspetto che l’interfaccia utente dovrà mantenere durante la creazione del dataset: essa sarà infatti continuamente cambiata nelle dimensioni e nel contenuto, venendo di fatto distrutta e ricostruita dopo la generazione di ogni immagine, e *CustomSkin.guiskin* funge, a tal proposito, da lista di parametri per la realizzazione di una nuova GUI, pronta a contenere un’altra immagine.

***Assets/Images***

La cartella analizzata in questo paragrafo contiene tutto il materiale necessario per la rappresentazione di una bounding box: quest’ultima sarà infatti formata dal connubio tra i file *Box.png* (riquadro verde)e *GlowOutline.png* (illuminazione bordi del riquadro) e sarà ridimensionata in funzione dei limiti spaziali dell’oggetto (o degli oggetti) su cui si basa il dataset, andando a fornire un’informazione sulla porzione di immagine all’interno della quale è presente ciò che ci interessa riconoscere.

***Assets/Materials***

Questa subdirectory contiene il file *backgroundMat.mat*, ossia il materiale che ricoprirà il piano frontale alla telecamera, operante da sfondo per ogni immagine del dataset.

L’immagine che il file “spalma” sul piano di background cambierà ad ogni iterazione del processo di generazione e sarà pescata dalla cartella *Assets/Resources/Textures*, il cui contenuto è randomizzato nell’ordine all’inizio di ogni ciclo di generazione.

***Assets/Resources***

La cartella *Assets/Resources* contiene la directory *Textures*, all’interno della quale sono poste tutte le fotografie che fungeranno da background per le immagini del dataset che verrà generato.

Le immagini presenti in *Textures* sono state scaricate mediante l’estensione Chrome chiamata Fatkun Batch Downloader e raffigurano, perlopiù, l’interno di abitazioni.  
La scelta di utilizzare immagini con questo specifico soggetto è dettata dal fatto che l’efficacia del detector allenato con questo dataset verrà verificata mediante le immagini provenienti da una webcam posta all’interno di un’abitazione: si è quindi cercato di rendere il dataset il più rappresentativo possibile della situazione reale all’interno della quale si sarebbe effettuata, almeno durante la fase di testing, la detection.

***Assets/Scripts***

La cartella analizzata in questo paragrafo contiene tutto ciò che regola e automatizza il processo di generazione del dataset. Per capire perfettamente come questo avviene, verrà fatta una breve descrizione di ogni script C# presente in questa cartella.

Interfacce:

* *IChangeable.cs*:Interfaccia implementata da tutte le classi il cui nome contiene la parola “*Change*”, impone a queste ultime di realizzare una funzione senza ritorni chiamata *ChangeRandom()*;

Classi astratte:

* *Singleton.cs*:  
  Classe che implementa il pattern Singleton[[4]](#footnote-4), garantendo a tutte le classi che la ereditano un’unica istanza e un singolo punto d’accesso ad essa tramite la proprietà *Instance* e il metodovirtuale *Awake()*. Verrà ereditata solo da *RandoTextures.cs*;

Classi (eccetto *RandoTextures.cs*, sono tutte classe figlie di *MonoBehaviour* e si inquadrano quindi come script strettamente pensati per essere collegati a oggetti presenti in una scena Unity 3D):

* *ChangeAmbient.cs*:Cambia la luce ambientale (ossia non proveniente da una fonte specifica ma diffusa su tutto lo spazio 3D) della scena Unity, assegnandole una tonalità randomica di bianco, tramite la funzione *ChangeRandom()* e la riporta alla tonalità predefinita a fine ciclo di generazione;
* *RandoTextures.cs*:  
  Unica classe utility, ridefinisce *Awake()* della classe *Singleton* da cui deriva, aggiungendo un mescolamento delle immagini presenti in *Assets/Resources/Textures* al momento della prima chiamata tramite la funzione *Shuffle()*.  
  Tali immagini vengono poi passate, una per ogni invocazione, tramite il metodo *GetRandomTexture(),* nell’ordine in cui si trovano dopo essere state mescolate;
* *ChangeTexture.cs*:  
  Cambia la fotografia che farà da sfondo all’immagine generata facendosela passare da *RandoTextures.cs* tramite l’implementazione di *ChangeRandom()*;
* *ChangeTransform*.cs:  
  Cambia posizione e rotazione dell’oggetto al quale è assegnata tramite l’implementazione di *ChangeRandom()*.  
  Le coordinate spaziali sono scelte randomicamente all’interno di un range di valori che assicurino la visibilità dell’oggetto alla telecamera (e quindi la sua presenza all’interno dell’immagine).  
  Per quanto riguarda la rotazione, sebbene idealmente ci si vorrebbe assicurare una detection a 360 gradi dell’oggetto, si è scelto di imporre una rotazione massima di +/-20 gradi in quanto, con +/- 180 gradi, il dataset risultava poco allenante, dato il numero molto più ampio di posizioni assumibili dall’oggetto, spesso molto diverse tra loro e quindi poco digeribili da parte del detector, che non riusciva a associarle.  
  Traslando questo discorso all’hand detection, quanto scritto implica che il detector riconoscerà solo mani con il palmo rivolto verso la telecamera;
* *ChangeWindow*.*cs*:  
  Questa classe mette a disposizione un’implementazione di *ChangeRandom()* che, ricevendo l’oggetto relativo alla finestra di esecuzione di Unity dalla funzione *GetMainGameView()*, la ridimensiona, ridimensionando di pari passo l’immagine, che andrà a finire nel dataset, contenuta in tale finestra.

1. https://github.com/MatthewHallberg/SyntheticDataGenerator [↑](#footnote-ref-1)
2. Verranno ignorate le altre cartelle visibili all’interno della directory di progetto *SyntheticDataGenerator*, in quanto contenenti file comuni a qualsiasi soluzione Unity, fatta eccezione per *TFUtils,* che verrà però descritta nei paragrafi inerenti alla fase di training [↑](#footnote-ref-2)
3. https://free3d.com/3d-model/freerealsichand-85561.html [↑](#footnote-ref-3)
4. Pattern che fa parte della “Gang of Four”, descritta nel libro “Design patterns” che fornisce patterns utilizzabili nell’OOP [↑](#footnote-ref-4)