POLITECNICO DI TORINO

Laurea Magistrale in INGEGNERIA BIOMEDICA

Tesi di Laurea Magistrale

Tracciamento markerless chirurgico: pipeline per la generazione di dati sintetici per la stima della posa di strumenti chirurgici

Relatori

Prof. Luca ULRICH

PhD. Federico SALERNO

Candidato

Samuele RASETTO

DICEMBRE 2025

Tracciamento markerless chirurgico: pipeline per la generazione di dati sintetici per la stima della posa di strumenti chirurgici

Samuele Rasetto

Abstract

Negli ultimi anni le tecniche di tracciamento ottico, il cui obiettivo è la localizzazione tridimensionale tramite sensori ottici per stimare la posizione e l'orientamento di oggetti nello spazio, hanno rappresentato un pilastro fondamentale in molteplici ambiti della Computer Vision. In particolare, la Pose Estimation markerless (PE-M) si configura come l'evoluzione naturale dei metodi marker-based tradizionali: essa permette di stimare la posizione e l'orientamento di un oggetto direttamente dalle sue caratteristiche visive, eliminando la necessità di marker fisici o rig di tracciamento; aspetto fondamentale nei contesti dove la loro introduzione risulta invasiva o poco pratica, come in sala operatoria. Le applicazioni della PE-M spaziano dalla manipolazione robotica alla guida autonoma, fino al settore biomedicale, dove la PE-M di strumenti chirurgici e strutture anatomiche abilita nuove forme di navigazione intraoperatoria, supervisione dei gesti e valutazione delle abilità chirurgiche, riducendo significativamente la complessità e l'ingombro dei setup rispetto alla controparte marker-based. Tuttavia, l'addestramento di modelli di deep learning per PE-M richiede un'enorme quantità di dati annotati, la cui acquisizione in ambienti reali risulta complessa e costosa. Questo lavoro si propone di sviluppare una pipeline end-to-end per la generazione di dati sintetici e la PE-M di strumenti chirurgici in ambiente Unity. La pipeline consente di produrre automaticamente immagini RGB, maschere di segmentazione, mappe di profondità e annotazioni 2D/3D, includendo keypoints e vertici delle bounding box tridimensionali. Sono stati modellati tre strumenti chirurgici rappresentativi: trapano, uno strumento di taglio e un puntatore marker-based; successivamente renderizzati in scenari sintetici con domain randomization di sfondi, illuminazione, materiali e posizionamento, così da aumentare la robustezza del modello e ridurre il domain gap rispetto ai dati reali. Il dataset sintetico generato è stato impiegato per l'addestramento di PVNet (Pixel-wise Voting Network), un modello di tipo instance-level, ossia progettato per riconoscere una specifica istanza di oggetto con elevata accuratezza. PVNet predice campi vettoriali direzionali verso i keypoints di interesse e calcola la posa tramite algoritmo Perspective-n-Point. L'addestramento è stato condotto su diverse configurazioni di iperparametri e varianti del dataset per individuare il miglior compromesso tra accuratezza e stabilità numerica, ottenendo alla fine un modello per la predizione di keypoints e uno per la predizione dei vertici della bounding box 3D. La validazione del metodo su dati reali, che prevede l'utilizzo di repliche fisiche degli strumenti stampate in 3D e una pipeline per la generazione delle annotazioni, costituisce la fase conclusiva del progetto ed è volta a quantificare l'effettivo trasferimento di conoscenza dal dominio sintetico a quello reale. I risultati

preliminari evidenziano la validità dell'approccio proposto nella generazione automatizzata di dataset fotorealistici e nell'addestramento di modelli di PE-M basati su deep learning, confermando il potenziale dei dati sintetici come risorsa scalabile per applicazioni chirurgiche di realtà aumentata. In prospettiva, l'estensione del sistema a un numero maggiore di strumenti e l'integrazione in piattaforme AR per il supporto intraoperatorio potranno rappresentare un passo significativo verso un tracciamento markerless robusto e clinicamente applicabile.

RINGRAZIAMENTI

Desidero esprimere la mia sincera gratitudine alle persone e agli enti che mi hanno permesso di giungere fino a questo traguardo. Ringrazio il mio relatore, Luca Ulrich, per avermi offerto l'opportunità di intraprendere questo percorso e per la costante disponibilità dimostrata, tanto nell'ambito del progetto quanto al di fuori di esso. Un ringraziamento particolare va al dottorando Federico Salerno, che mi ha seguito nei mesi passati, per il suo supporto nel superare le difficoltà incontrate, nel fornirmi il materiale necessario, nel revisionare i miei elaborati e, più in generale, per la sua guida attenta e competente. Ringrazio inoltre il 3DLAB, che mi ha messo a disposizione i suoi spazi e le sue attrezzature. Infine, estendo un sentito ringraziamento a tutte le persone che, in modo consapevole o meno, mi hanno sostenuto lungo questo percorso.

Indice

Elenco delle figure

Elenco delle tabelle

Acronimi

ADD	Average Distance of Model Points.
ADD-S	Avarage Closest Point Distance.
AR	Augmented Reality.
BB-PVNet	BoundingBox-Pixel-wise Voting Network.
BOP	Benchmark for 6D Object Pose Estimation.
CV	Computer Vision.
DL	Deep Learning.
DoF	Degrees Of Freedom.
GAN	Generative Adversarial Networks.
GT	Ground Truth.
HMD	Head-Mounted Display.
KP-PVNet	Keypoints-Pixel-wise Voting Network.
MDE	Mean Distance Error.
SOLO	Synthetic Optimized Labeled Objects.
SOTA	State of the Art.
PCK	Percentage of Correct Keypoints.
PE-M	Pose Estimation Markerless.
PnP	Perspective-n-Point.
PVNet	Pixel-wise Voting Network.
VR	Virtual Reality.

Capitolo 1

Introduzione

1.1 Background

Nell'ultimo decennio, il tracking di oggetti in tempo reale è diventato oggetto di numerosi studi in ambito di Computer Vision (CV), con applicazioni che spaziano dalla robotica alla realtà aumentata. All'interno di questo ampio dominio, si possono distinguere due approcci principali: il marker-based tracking, che si basa su elementi fiduciali noti per stimare la posa di un oggetto, e il markerless tracking, che opera direttamente sulle caratteristiche visive dell'oggetto senza necessità di modificare la scena con elementi artificiali. Il markerless tracking rappresenta una sfida più complessa, in quanto richiede l'identificazione e il tracking dell'oggetto basandosi esclusivamente sulle sue proprietà intrinseche come forma, texture e geometria, in aggiunta l'assenza di marker è necessaria per scenari reali dove il loro utilizzo non è praticabile, come in ambiti industriali o chirurgici. Si differenzia in task di tracking più semplici come il 2D object tracking, che si limita a seguire una bounding box nel piano immagine, o in tracking più complessi che considerano la posizione dell'oggetto nello spazio 3D.

All'interno del paradigma markerless, la Pose Estimation Markerless (PE.M) si configura come il problema di stimare simultaneamente la posizione tridimensionale, cioè la traslazione lungo gli assi X, Y, Z, e l'orientazione, quindi la rotazione attorno agli stessi assi, di un oggetto rispetto al sistema di riferimento della camera. In base all'applicazione, la PE-M può essere condotto su un diverso numero di gradi di libertà (DoF):

- 3DoF, si studia solo la rotazione dell'oggetto;
- 6DoF, si studia la rotazione e la traslazione;
- 9DoF, si studia la rotazione, la traslazione e la dimensione dell'oggetto.

Formalmente, la 6DoF PE può essere rappresentata come una trasformazione rigida composta da una matrice di rotazione R e un vettore di traslazione t. Un'analisi matematica del problema più approfondita sarà fornita nel prossimo capitolo.

Le applicazioni di maggiore interesse della PE-M si ritrovano in diversi ambiti della CV. Un primo settore è quello della guida autonoma, dove tali tecniche vengono

utilizzate per analizzare l'ambiente circostante e identificare possibili ostacoli per il veicolo ([1],[29] [55]). Un secondo ambito è la 6DoF *Grasp Estimation* in robotica, ossia la PE-M dell'oggetto per coordinare un braccio meccanico incaricato di afferrarlo ([79], [45], [6], [77]). Infine, un ulteriore settore applicativo è quello della realtà aumentata (AR) e della realtà virtuale (VR), generalmente tramite overlay sovrapposti alla visuale dell'utente e mediante *Head-Mounted Display* (HMD).

La coniugazione tra queste tecniche di CV e l'ambiente chirurgico è oggetto di crescente attenzione. In sala operatoria, l'impiego della AR riguarda in particolare il riconoscimento e il tracciamento, marker-based o markerless, nello spazio 3D di strumenti chirurgici e strutture anatomiche, con l'obiettivo di migliorare l'outcome dell'intervento [42], agevolare il chirurgo quando opera in condizioni di visibilità ridotta e mobilità limitata [76], e supportare l'addestramento di nuovi chirurghi attraverso l'analisi combinata dei movimenti della mano e del tracciamento degli strumenti e lo *skill assessment*, ossia la valutazione della precisione e della fluidità dei gesti chirurgici [17]. A queste applicazioni si affiancano inoltre la navigazione intra-operatoria, che consente la fusione di immagini pre-operatorie con il campo operatorio reale e la possibilità di aggiungere sull'interfaccia dell'utente dati riguardanti il paziente e feedback real-time dell'operazione. [68] [43] [13] [70]

Recenti studi clinici dimostrano che l'impiego della AR in chirurgia può migliorare significativamente l'accuratezza procedurale, con tassi di precisione del 93-100% in chirurgia spinale e ortopedica.[10] [4] [39] Il mercato globale della AR in ambito sanitario è previsto crescere da 3,4 miliardi di dollari nel 2023 a oltre 11 miliardi entro il 2030, riflettendo la crescente adozione clinica di queste tecnologie. [20]

1.2 Contributo

L'obiettivo di questo lavoro è presentare e analizzare due metodi di pose estimation per strumenti chirurgici addestrati su un dataset sintetico, generato attraverso una pipeline sviluppata in ambiente Unity. Il primo metodo si focalizza sulla predizione dei keypoints posti sugli strumenti, mentre il secondo predice i vertici della bounding box 3D, in entrambi i casi la posa viene ricavata a posteriori risolvendo il problema dei Perspective-n-Point. L'elaborato si articola in cinque capitoli. Nel Cap.1 vengono introdotti i concetti fondamentali e lo stato dell'arte relativi alla pose estimation e alla generazione di dati sintetici, con un'attenzione particolare ai metodi e alle tecnologie più rilevanti presenti in letteratura. Il Cap.2 è dedicato ad approfondire i concetti teorici della PE-M. Cap.3 si concentra sulla descrizione della pipeline di generazione implementata in ambiente Unity, illustrandone le scelte progettuali e le modalità di creazione del dataset, nonché l'addestramento dei modelli selezionati e la sperimentazione in laboratorio volta a valutarne le prestazioni su dati reali, i risultati di questi modelli vengono mostrati nel Cap.4. Infine, il Cap.5 è incentrato sull'analisi dei risultati ottenuti, mettendo in relazione le performance con le scelte progettuali adottate, e discute la validità del dataset sintetico come strumento per

l'addestramento di modelli applicabili a scenari reali, evidenziando al contempo i limiti emersi e le possibili linee di sviluppo futuro del progetto.

1.3 Stato dell'arte per pose estimation

Come mostra la Fig. 1.1, estrapolata dalla ricerca effettuata da Hoque et al. (2021), nell'ultimo decennio il numero di studi condotti su questa tematica è aumentato esponenzialmente. Prima dell'avvento del Deep Learning (DL), i metodi tradizionali per la stima della posa si basavano principalmente su tecniche di feature matching: approcci classici includevano l'uso di feature come Scale-Invariant Feature Transform (SIFT) [38], Fast Point Feature Histograms (FPFH) [53], ossia descrittori geometrici utilizzati per stabilire corrispondenze tra immagini 2D e modelli 3D, che tuttavia si dimostrarono poco efficaci con oggetti senza texture o molto complessi. [49] [78] Con l'evoluzione del DL anche i metodi 6DoF sono migliorati, verranno ora analizzate le tecniche allo stato dell'arte (SOTA), ricercate a partire dal 2015, per 6DoF PE suddivise nelle due principali macro-categorie. [35]

1.3.1 Instance-level object pose estimation

Fanno parte di questa categoria quei metodi che vengono allenati per riconoscere una specifica istanza di un oggetto, quindi hanno performance molto elevate per quella specifica istanza, ma peggiorano con scene più complesse, perdendo la capacità di generalizzare. La stima può essere effettuata grazie all'identificazione di keypoints tramite voting o regressione, e l'analisi di come questi punti hanno cambiato posizione da un frame rispetto al modello CAD originale. Per fare ciò bisogna trovare anzitutto le corrispondenze dal mondo 3D al mondo 2D, quindi trovare i punti nello spazio immagine, dopodiché tramite algoritmi come il Perspective-n-Point (PnP) si può ricavare la posa della camera. Oppure tramite tecniche di template matching, ossia si cerca di far combaciare il modello in esame con una serie di template generati artificialmente tramite l'analisi delle features e il modello che combacia maggiormente sarà scelto per l'identificazione di translazione e rotazione.

Peng et al. nel 2019 ha proposto PVNet (Pixel-wise Voting Network), uno dei primi modelli appartenenti a questa categoria. Il principio è quello di non regredire direttamente le coordinate dei punti, bensì predire dei vettori direzionali che puntano nella direzione dei keypoints, dopodiché tramite un processo di votazione basato su RANSAC vengono create delle distribuzioni spaziali di probabilità. La posa viene successivamente calcolata con un algoritmo PnP uncertainty-driven, cioè che tiene conto del fatto che la posizione predetta è incerta. Il modello ha performato ottimamente sul dataset LINEMOD, con una ADD = 86.3%. [48]

Li et al. (2019) introduce Coordinates-based Disentangled Pose Network (CDPN), che separa la stima della traslazione e della rotazione, trattandoli come due problemi distinti. La prima viene ricavata indirettamente tramite coordinate dense 3D per ogni pixel dell'oggetto in combinazione con PnP, mentre la seconda viene regredita

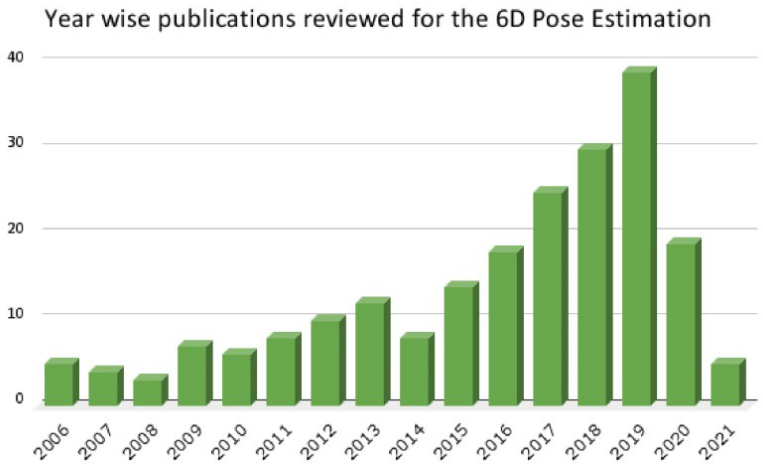


Figura 1.1: Numero di articoli pubblicazioni
Il grafico mostra il numero di articoli pubblicati dal 2006 al 2021 relativi a 6DoF pose estimation. Si osserva un aumento significativo, segno del crescente interesse verso la tematica.[29]

direttamente dall'immagine RGB. Lo studio dimostra stime accurate su oggetti texture-less e occlusi, risolvendo i principali problemi esposti per i metodi feature-based. [34]

Nel 2019 Wang et al. creano DenseFusion, un framework che utilizza immagini RGB-D, quindi immagini RGB con l'informazione aggiuntiva sulla profondità; processa separatamente le due sorgenti di dati e poi le fonde per estrarre *feature embedding dense pixel-wise*. Per ogni pixel viene stimata una posa e le predizioni vengono successivamente aggregate tramite voting. Il modello sfrutta una procedura end-to-end capace di fare inferenza quasi in real-time, senza bisogno di post-processing, come PnP. Raggiunge performance allo stato dell'arte su YCB e LineMOD. [71]

He et al. nel 2020 estende l'idea di PVNet, creando un modello che lavora su immagini RGB-D e sfrutta informazioni geometriche 3D. La rete usa come backbone PointNet++ per l'estrazione di features da nuvole di punti e per ognuno di essi predice i vettori direzionali che puntano verso i keypoints. Quest'ultimi vengono localizzati tramite clustering dei voti e la posa viene ricavata tramite algoritmo PnP. Il modello ha raggiunto ADD = 91.8% su LineMOD e 52.6% su YCB-Video. [23]

Una delle prime applicazioni del template matching è stata proposta da Payet & Todorovic (2011), i contorni dell'oggetto vengono utilizzati come feature principale per la detection e la pose estimation. Partendo da un modello CAD come input, viene generata una serie di rappresentazioni 2D del contorno dell'oggetto da diverse angolazioni, successivamente tramite un matching gerarchico viene scelto il miglior template ottenendo immediatamente la rotazione; la traslazione viene poi raffinata allineando il template alla posizione dell'oggetto nell'immagine. [47]

1.3.2 Category-level object pose estimation

Questi metodi imparano a generalizzare meglio dei metodi basati su istanze, ma richiedono una grande mole di dati per il training su una categoria poiché devono

andare a stimare la posa di oggetti mai visti prima appartenenti a categorie conosciute. Si differenziano in due categorie: quelli che si basano su rappresentazioni geometriche che catturano la forma tipica della categoria, 'shape prior-based', acquisite da modelli CAD, e quelli che si basano solo su feature geometriche e semantiche senza bisogna della forma, shape prior-free.

Il lavoro più significativo è stato quello di Wang et al. (2019), che ha introdotto la stima per categorie. Normalized Object Coordinate Space (NOCS) propone uno spazio di coordinate normalizzate che fornisce una rappresentazione unificata per tutti gli oggetti di una determinata categoria. Per ogni pixel vengono predette le coordinate NOCS, viene poi predetta una maschera di segmentazione binaria e la classe dell'oggetto, tramite l'algoritmo di Umeyama viene allineata la nuvola di punti dell'oggetto con la NOCS map da cui poi si può ricavare la posa. [69] [73]

6D-ViT usa come input immagini RGB-D e il framework è suddiviso in due Vision Transformer, Pixelformer e Pointformer, che si occupano di estrarre rappresentazioni pixel-wise dell'oggetto e di acquisire le feature point-wise della nuvola di punti. Poi fonde queste informazioni e crea la NOCS map che tramite la trasformazione di similitudi con la nuvola di punti permette di regredire la posa finale. [82]

Di et al. (2022) [8] propone un metodo category level basato sulla geometria dell'oggetto, quindi shape prior-free, chiamata GPV-Pose (Pose Estimation via Geometry-guided Point-wise Voting). Viene innanzitutto creata la nuvola di punti sull'oggetto anche tramite l'utilizzo delle simmetrie e durante il training ogni punto vota per i vertici della 3D bounding box con un apprendimento guidato dalle informazione geometriche. Il test su REAL275 raggiunge 79.1% di IoU superando i metodi shape prior-based.

Liu et al. (2023) [36] presenta IST-Net, un modello shape prior-free che serve a dimostrare che la ricostruzione deformativa è più importante dello shape-prior stesso. Basa il suo funzionamento su una trasformazione di spazio implicito, ossia lo spazio delle features viene trasformato invece di essere deformato per ottenere una rappresentazione implicita che cattura la variazione intra-classe della categoria. Questo studio apre un nuovo filone di ricerca sui metodi prior-free, mettendo in discussione l'utilità degli shape priors.

1.4 Dati reali versus dati sintetici

Il crescente utilizzo di metodi basati sul DL richiede massicce quantità di dati per l'addestramento. Per garantire che la rete utilizzata riesca a generalizzare bene conviene allenarla su dati reali, però l'acquisizione e l'etichettatura di questi può essere un processo laborioso e costoso. [59]

Illustriamo innanzitutto le metodologie per acquisire il ground truth in applicazioni di pose estimation. A differenza di task come detection o segmentazione, definire la rotazione 6D di un oggetto nello spazio non può essere svolta tramite semplice analisi visiva e richiede strumentazione specializzata. Il metodo più preciso prevede l'utilizzo di un braccio robotico calibrato, come il KUKA o l'UR5, con l'oggetto montato su di

esso, questo procedimento permette di ottenere accuratezze sub-millimetriche, ma il setup è costoso e non scalabile. Dataset come YCB-Video e T-Less hanno adottato questo metodo.

Un'alternativa alla robotica sono i sistemi di motion capture, come OptiTrack e Vicon, combinati all'utilizzo di marker riflettenti posti sull'oggetto. Grazie a una serie di telecamere infrarosse calibrate che triangolano i marker si può ricavare la posa anche in tempo reale. La scelta di un approcio marker-based però altera la struttura dell'oggetto e l'hardware ha un costo molto elevato. [19]

Un approccio alternativo utilizza la proiezione di pattern strutturati di luce per la ricostruzione 3D densa della scena, sfruttando setup stereo multi-camera. Questo metodo offre accuratezze nell'ordine di 1-2mm.

Una soluzione più economica prevede la scansione 3D dell'oggetto tramite scanner strutturato, seguita dall'allineamento manuale del modello 3D al frame RGB-D. Successivamente, algoritmi come Iterative Closest Point (ICP) affinano automaticamente la stima. [51]

Il vantaggio economico si traduce però in un notevole dispendio di manodopera e in accuratezze inferiori a causa dell'errore umano nell'inizializzazione.

I dataset sintetici, quindi simulazioni renderizzate di scene contenenti gli strumenti di interesse, cercano di sopperire a queste limitazioni. Una volta creato il motore di generazione aumenta esponenzialmente il quantitativo di dati e annotazioni che si possono generare per unità di tempo a costo quasi zero. Il fatto di utilizzare software di rendering permette di generare annotazioni perfette per qualsiasi task, come PE, maschere, profondità o determinazione dei keypoints 2D/3D. La possibilità di controllare completamente la scena permette di catturare immagini difficilmente riproducibili in condizioni reali, variando sistematicamente illuminazione, viewpoint della camera, disposizione spaziale degli oggetti e applicando *domain randomization* su texture e materiali.

L'utilizzo di questi dati introduce però criticità. Il principale problema è il *domain gap*: anche con rendering fotorealistici, sottili differenze nella fisica dei materiali, come riflettanza, trasparenza, rumore del sensore e nelle imperfezioni degli oggetti reali rendono la generalizzazione al mondo reale problematica. Questo si traduce nel rischio di overfitting al dominio sintetico, dove la rete mostra prestazioni elevate sui dati di training, ma degrada significativamente su dati reali. Inoltre, la creazione iniziale della pipeline di generazione richiede un investimento temporale non trascurabile per il design della scena e il tuning dei parametri di rendering.

Però queste limitazioni possono essere attenuate tramite una serie di accorgimenti in fase di addestramento. Primo fra tutti è i a *domain randomization* [63], ossia la variazione massiva di parametri come rumore, illuminazione e texture con il fine di rendere il mondo sintetico complesso come il mondo reale; ciò si può effettuare ad esempio tramite texture procedurali random oppure distorsioni ottiche. [65]

Un'altra possibilità è quella di effettuare *transfer learning*, ossia modelli pre-addestrati su un dataset sintetico passano per un processo di fine-tuning su dati reali, come è stato fatto da Wang et al. con il passaggio da NOCS-CAMERA a NOCS-

REAL. [73] Infine un terzo approccio utilizzato è quello di fare *domain adapation* cercando di allineare le distribuzioni delle feature estratte dal dominio sorgente (dati sintetici) con quelle del dominio target (dati reali). [14] [67]

Questo adattamento può avvenire in due modi:

1. Adattamento non supervisionato, l'obiettivo è apprendere una rappresentazione comune che minimizzi la discrepanza tra le distribuzioni delle feature dei due domini [37]; approcci classici usano Maximum Mean Discrepancy (MMD) per minimizzare la distanza tra le distribuzioni dei due domini nello spazio delle feature, oppure tramite Correlation Alignment (CORAL) [60] allineando le statistiche del secondo ordine (matrici di covarianza) delle feature tra dominio sorgente e target.

2. Adattamento avversario, sfrutta un'architettura a due componenti: un estrattore di feature ricava delle rappresentazioni dall'immagine di input e un discriminatore di domini, cioè un classificatore binario che tenta di distinguere se le feature provengono dal dominio sintetico o reale. Durante l'addestramento, la prima componente è ottimizzata in modo avversario per confondere il discriminatore. Un metodo molto comune è la CycleGAN che apprende una mappatura bidirezionale tra domini attraverso GANs cicliche, trasformando immagini sintetiche in stile fotografico mantenendo il contenuto semantico. [81] [46]

   Recentemente, [30] ha dimostrato che applicare la traduzione di stile a livello di singolo oggetto, piuttosto che sull'intera scena, migliora significativamente la preservazione delle geometrie e delle pose.

1.5 Stato dell'arte per la generazione di dati sintetici

Questa sezione analizza lo stato dell'arte delle pipeline di generazione sintetica, con particolare focus su Unity Perception, il framework utilizzato in questo lavoro. Come evidenziato da Schieber et al. [56] , i metodi di generazione sintetica possono essere divisi in quattro approcci principali:

- **Crop-out**, rappresentano l'approccio più semplice, combinando ritagli di oggetti reali su immagini di sfondo. Sebbene economici e veloci da implementare, questi metodi sono limitati dalla disponibilità di immagini dell'oggetto da diverse angolazioni e offrono opzioni minime di ground truth [18, 54].

- **Graphic API-based**, Gli approcci basati su API grafiche come OpenGL utilizzano modelli CAD 3D per il rendering su immagini provenienti da altri dataset [25, 41]. NViSII [44] rappresenta lo stato dell'arte in questa categoria, utilizzando NVIDIA OptiX per il ray tracing, ossia il calcolo dei percorsi effettuati dalla luce quando interagisce con delle superfici, permette di ottenere alti livelli di fotorealismo. Supporta una vasta gamma di opzioni di ground truth

e domain randomization, tuttavia richiede competenze in programmazione di basso livello e setup più complessi rispetto a framework basati su game engine.

- **3D Modeling-based**, vengono usati software di modellazione 3D, in particolare BlenderProc, un API basata su Blender e Python. Permette di caricare asset da dataset esterni, simulare la fisica tramite Bullet Physics, offrire un elevato fotorealismo e fornire output in diversi formattazioni. Gli svantaggi principali sono i tempi di rendering elevati e la necessità di conoscere l'ecosistema Blender. [7]

- **3D Game Engine-based**, offrono il miglior compromesso tra fotorealismo, velocità di rendering e facilità d'uso grazie a editor visuali integrati. I più utilizzati sono NDDS (NVIDIA Deep Learning Data Synthesizer) [62], un plugin per Unreal Engine, UnrealROX+ [40], che estende Unreal alle annotazione per gli hand joints a discapito da un framework per domain randomization, e infine Unity Perception [2], che verrà ora analizzato più nel dettaglio.

1.5.1 Unity Perception

Unity Perception [2] è un pacchetto open-source per Unity Engine progettato specificamente per la generazione di dati sintetici destinati ad applicazioni di computer vision. L'architettura si articola in tre componenti principali:

1. **Perception Camera**: componente che si attacca a qualsiasi camera Unity e gestisce la cattura automatica di annotazioni ground truth, incluse segmentazione semantica e di istanze, bounding box 2D/3D, keypoint 2D/3D configurabili e mappe di profondità.

2. **Randomizers**: sistema modulare per la domain randomization che opera su molteplici parametri della scena, tra cui proprietà delle fonti luminose (intensità, colore, tipologia), texture e materiali degli oggetti, pose (posizione e rotazione) di oggetti e camera. Nuovi randomizer personalizzati sono implementabili tramite scripting in C#.

3. **Simulation Scenario**: container che orchestra la sequenza di randomizzazione e cattura dati. Gestisce il numero di iterazioni configurabili, coordina l'esecuzione di tutti i randomizer registrati e controlla il sampling dei parametri secondo distribuzioni specificate (uniforme, normale, categorica).

Il numero di frame generati per unità di tempo di Unity Perception risulta significativamente superiore rispetto a BlenderProc grazie all'utilizzo di rasterization real-time invece di path tracing, rendendo la produzione di dataset su larga scala più efficiente dal punto di vista computazionale. Il framework beneficia dell'ecosistema Unity, offrendo un workflow WYSIWYG (What You See Is What You Get) che facilita la costruzione di scene complesse tramite interfaccia drag-and-drop, preview real-time del risultato finale, debugging visuale dei parametri di randomizzazione e

inspector per la modifica dei parametri senza necessità di scripting. L'integrazione con l'ecosistema .NET permette inoltre l'utilizzo di librerie esterne per estendere le funzionalità del framework.

Il sistema di domain randomization di Unity Perception si distingue per la sua configurabilità: supporta sampling parametrico con seed riproducibili per garantire esperimenti replicabili, scheduling temporale dei randomizer per controllare quali parametri variare in specifiche iterazioni, e randomizzazione selettiva basata su tag per applicare trasformazioni solo a sottoinsiemi di oggetti nella scena.

Per quanto riguarda il rendering, la High Definition Render Pipeline (HDRP) con supporto hardware-accelerated ray tracing (DXR) offre un compromesso equilibrato tra qualità fotorealistica e velocità di generazione, sebbene non raggiunga il livello di fotorealismo ottenibile con Cycles renderer di BlenderProc o il path tracing di NVIDIA Isaac Sim in configurazioni senza accelerazione hardware.

Tuttavia, Unity Perception presenta alcune limitazioni. Il motore fisico PhysX, ottimizzato per simulazioni real-time in contesti gaming, non fornisce la stessa accuratezza e stabilità del Bullet Physics Engine utilizzato da BlenderProc, risultando meno affidabile per scene con elevata complessità fisica. L'implementazione di randomizer personalizzati richiede competenze in C# e familiarità con l'architettura Unity, rappresentando una barriera d'ingresso per ricercatori abituati a workflow basati su Python. Inoltre, l'assenza di librerie Python native per computer vision (NumPy, OpenCV, SciPy) rende il post-processing dei frame generati meno immediato.

1.6 Dataset per 6D Object Pose Estimation

Verranno quindi mostrati alcuni dei dataset più utilizzati nell'ambito della pose estimation, proposti dal Benchmark for 6D Object Pose Estimation (BOP) [27],il benchmark di riferimento per questo ambito. I dataset sono categorizzati in base alla natura dei dati: reali, sintetici o misti.

1.6.1 Dataset reali

I dataset reali sono acquisiti tramite sensori RGB-D in scenari controllati, con annotazioni di ground truth ottenute mediante setup robotici calibrati, sistemi di motion capture o annotazione manuale seguita da refinement con algoritmi di registrazione.

LINEMOD (LM) Il dataset LINEMOD [24] comprende 15 sequenze di oggetti prevalentemente textureless, contenenti immagini RGB-D annotate con pose 6DoF, modelli CAD, bounding box 2D e maschere binarie. Le sequenze presentano scenari con clutter, oggetti privi di texture distintive e condizioni di illuminazione variabili. Ogni sequenza contiene circa 1.200 frame con un singolo oggetto visibile per frame.

LINEMOD Occlusion (LM-O) Estensione del dataset LM introdotta da Brachmann et al. [3] , LM-O fornisce 1214 immagini RGB-D caratterizzate da molteplici istanze di oggetti con diversi livelli di occlusione reciproca. Questo dataset

viene usato per testare la capacità dei metodi di gestire occlusioni parziali, dove solo una porzione dell'oggetto è visibile.

YCB-Video (YCB-V) Introdotto da Xiang et al. [74], fornisce annotazioni di pose 6D per 21 classi di oggetti domestici comuni distribuiti su 92 video RGB-D, per un totale di 133827 frame. Il dataset è acquisito con sensore RealSense in ambienti indoor con illuminazione naturale e presenta scene cluttered con occlusioni realistiche. Le pose ground truth sono ottenute tramite setup robotico calibrato.

T-LESS Il dataset [28] si focalizza su 30 oggetti industriali textureless e privi di colori discriminanti, caratterizzati da simmetrie geometriche. Include circa 48000 immagini catturate con tre sensori sincronizzati (structured-light RGB-D, time-of-flight RGB-D e fotocamera RGB ad alta risoluzione) a risoluzioni variabili. Le annotazioni ground truth sono acquisite tramite setup robotico con accuratezza sub-millimetrica.

1.6.2 Dataset Sintetici

I dataset sintetici sono generati tramite rendering 3D di modelli CAD o mesh ricostruite, utilizzando motori di rendering fotorealistici o game engine. Offrono annotazioni perfette e scalabilità illimitata, ma soffrono del domain gap rispetto ai dati reali.

CAMERA (NOCS-CAMERA) Componente sintetica del dataset NOCS [72], contiene circa 300000 immagini RGB-D renderizzate di oggetti appartenenti a 6 categorie. Gli oggetti sono campionati da ShapeNetCore [5], garantendo elevata variabilità intra-categoria con circa 1000 modelli 3D per categoria. Oltre alle pose 6D, fornisce mappe NOCS per category-level pose estimation.

Synthetic HOPE Dataset sintetico introdotto da Shi et al. [57], contenente 60000 immagini per ciascuno dei 28 oggetti domestici scansionati con scanner 3D EinScan-SE. Generato utilizzando NDDS e NViSII [44], applica *domain randomization* a cinque livelli di complessità per illuminazione, occlusioni (via flying distractors) e variazioni di background.

FAT (Falling Things) Presentato da Tremblay et al. [64], FAT contiene 61.000 frame di 21 oggetti YCB renderizzati con NDDS in Unreal Engine. Sono presenti frame con il singolo oggetto isolato e scene multi-oggetto.

1.6.3 Dataset Misti

I dataset misti combinano dati sintetici per training/pre-training con subset reali per validazione e fine-tuning, sfruttando i vantaggi di entrambi gli approcci.

NOCS (CAMERA + REAL) Il dataset completo NOCS [72] combina:

- **CAMERA** (sintetico): 300K immagini per pre-training, come descritto sopra
- **REAL275**: 4300 scene reali catturate con Intel RealSense D415 in 7 scenari diversi, di cui 275 scene costituiscono il test set standard. Le annotazioni

ground truth sono ottenute tramite annotazione manuale iniziale seguita da ICP refinement, con modelli 3D degli oggetti reali acquisiti tramite scanning.

HomebrewedDB Dataset introdotto da Kaskman et al. [31], contenente 33 oggetti domestici comuni con 17420 frame reali RGB-D (13 scene) e 119000 frame sintetici generati con BlenderProc. Gli oggetti includono prodotti commerciali con texture variabili.

HOPE (Household Objects for Pose Estimation) Dataset presentato nel BOP Challenge 2020, HOPE [66] fornisce 28 oggetti domestici scansionati e 50 scene reali RGB-D con illuminazione variabile catturate in 10 ambienti diversi.

Capitolo 2

Fondamenti teorici

Questo capitolo fornisce una trattazione approfondita dei concetti teorici di base necessari alla comprensione delle trasformazioni geometriche e delle tecniche di stima della posa, che costituiscono il fondamento dell'elaborato.

2.1 Sistemi di riferimento e coordinate

Nella visione artificiale, è necessario lavorare con diversi sistemi di coordinate, ciascuno con un ruolo specifico.

Sistema di coordinate globale: È il sistema di riferimento globale e assoluto rispetto al quale vengono definite le posizioni degli oggetti nella scena tridimensionale. Le coordinate in questo sistema sono indicate come (X_w, Y_w, Z_w).

Sistema di coordinate camera: È un sistema solidale con la camera, con origine nel centro ottico della camera stessa. L'asse Z è tipicamente orientato lungo l'asse ottico principale, mentre X e Y giacciono sul piano perpendicolare. Le coordinate sono indicate come (X_c, Y_c, Z_c).

Sistema di coordinate immagine: Rappresenta il piano dell'immagine espresso in unità fisiche (millimetri). L'origine è solitamente posizionata nel punto principale, definito come intersezione dell'asse ottico con il piano immagine, con coordinate (x, y).

Sistema di coordinate pixel: È il sistema discreto dell'immagine digitale, con origine nell'angolo superiore sinistro e assi misurati in pixel. Le coordinate sono indicate come (u, v).

Questi concetti sono trattati in dettaglio in [21].

Secondo il modello *pinhole*, illustrato in Fig. 2.1, tutti i raggi luminosi provenienti dalla scena passano attraverso un unico punto, il centro di proiezione, prima di raggiungere il piano immagine. La relazione fondamentale della proiezione prospettica è data dalla similitudine dei triangoli che si formano tra piano immagine - centro di proiezione e tra centro di proiezione - sistema globale. Considerando un punto P = (X_c, Y_c, Z_c) nel sistema di coordinate camera e la sua proiezione p = (x, y) sul piano immagine posto a distanza focale f dal centro di proiezione, le equazioni per la

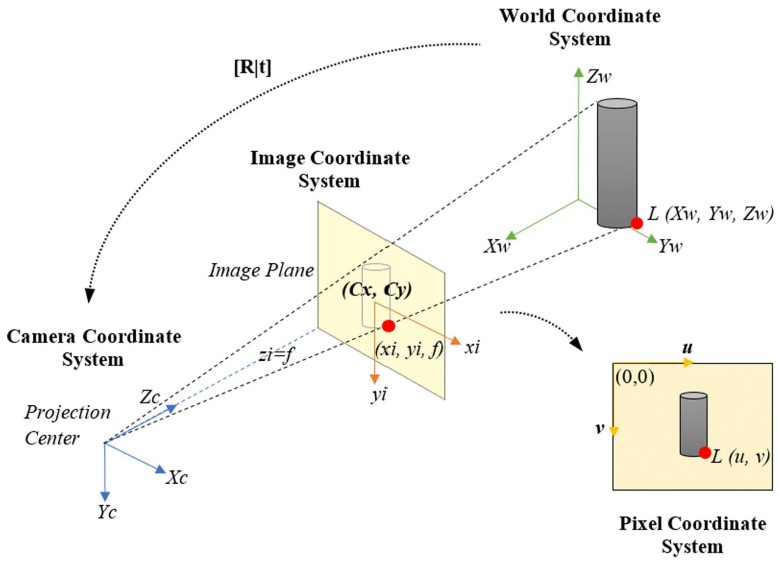


Figura 2.1: Schema del modello *pinhole*
Viene illustrato il passaggio dal sistema globale al sistema della camera tramite i parametri estrinseci definiti come rotazione R e traslazione t con il passaggio intermedio attraverso il piano immagine. Il sistema di coordinate immagine viene convertito nel sistema di coordinate in pixel tramite i parametri intrinseci della camera. f è la distanza focale tra piano immagine e camera, la retta passante per il centro del sistema di riferimento camera e il punto C_x, C_y, centro del piano immagine, è l'asse ottico principale. [11]

proiezione prospettica risultano:

$$x = f \cdot \frac{X_c}{Z_c} \tag{2.1}$$

$$y = f \cdot \frac{Y_c}{Z_c} \tag{2.2}$$

Per linearizzare le equazioni di proiezione e facilitare le composizioni di trasformazioni, si utilizzano le coordinate omogenee. Un punto bidimensionale (x, y) è rappresentato come un vettore tridimensionale $(x, y, 1)^T$, mentre un punto tridimensionale (X, Y, Z) diventa $(X, Y, Z, 1)^T$. In coordinate omogenee, la proiezione prospettica può essere espressa come:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.3}$$

dove:

$$\lambda = \frac{1}{Z_c} \tag{2.4}$$

è il fattore di scala. Andiamo ora a definire i parametri intrinseci della camera, ossia le caratteristiche interne ad essa che non dipendono dalla posizione nello spazio. La matrice di calibrazione K trasforma le coordinate nel sistema camera in coordinate

pixel:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

con:

$$f_x = f \cdot m_x \quad \text{e} \quad f_y = f \cdot m_y \tag{2.6}$$

dove f_x e f_y sono le lunghezze focali espresse in pixel lungo gli assi x e y, con m_x, m_y sono i fattori di scala pixel/millimetro, c_x e c_y rappresentano le coordinate del punto principale e idealmente equivalgono al centro dell'immagine e infine s è il parametro di skew che modella la non-ortogonalità degli assi dell'immagine. Nelle camere moderne è tipicamente trascurabile (s = 0). La calibrazione della camera per determinare questi parametri può essere effettuata con il metodo di Zhang [80], uno degli approcci più diffusi che utilizza un pattern planare (tipicamente una scacchiera) osservato da multiple viste.

I parametri estrinseci descrivono la posizione e l'orientamento della camera nel sistema di coordinate mondo. Sono composti da un vettore di traslazione **t** 3D che specifica la posizione del centro ottico della camera nel sistema mondo e da una matrice di rotazione **R** $3x3$ (9 parametri per 3 gradi di libertà) che descrive l'orientamento della camera. La trasformazione di un punto dal sistema mondo al sistema camera è data da:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t} \tag{2.7}$$

con

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.8}$$

Oltre alla rappresentazione matriciale la rotazione può essere espressa in angoli di Eulero, quindi tramite una sequenza di tre rotazioni elementari attorno agli assi del sistema di coordinate. La rappresentazione classica ZYX (yaw-pitch-roll) è:

$$R = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{2.9}$$

dove ψ, θ, ϕ sono gli angoli attorno agli assi Z, Y e X.

Vi è ancora una terza rappresentazione che utilizza i quaternioni $\mathbf{q} = [q_x, q_y, q_z, q_w]^T|$ che possono essere convertiti in matrice di rotazione **R** usando:

$$\mathbf{R} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & 1 - 2(q_x^2 + q_z^2) & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & 1 - 2(q_x^2 + q_y^2) \end{bmatrix} \tag{2.10}$$

oppure in gradi di Eulero tramite:

$$\begin{aligned} \text{Yaw } (\psi) &= \text{atan2}\left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)\right) \\ \text{Pitch } (\theta) &= \arcsin\left(\text{clamp}(2(q_w q_y - q_z q_x), -1, 1)\right) \\ \text{Roll } (\phi) &= \text{atan2}\left(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)\right) \end{aligned} \tag{2.11}$$

Per combinare rotazione e traslazione in un'unica operazione matriciale, si utilizza la rappresentazione in coordinate omogenee. La trasformazione di un punto diventa:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.12}$$

Combinando parametri intrinseci ed estrinseci, la proiezione di un punto 3D del mondo $P_w = (X_w, Y_w, Z_w, 1)^T$ nelle coordinate pixel $p = (u, v, 1)^T$ è data da:

$$\lambda \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \cdot \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.13}$$

La matrice di proiezione completa $P = K \cdot [R|t]$ è una matrice 3×4 che mappa direttamente coordinate mondo in coordinate immagine, a meno del fattore di scala.

2.2 Il problema PnP

Il problema Perspective-n-Point [12] consiste nel determinare la posa (R, t) della camera dati:

- n punti 3D nello spazio mondo $P_i = (X_i, Y_i, Z_i)$;
- i corrispondenti punti 2D nell'immagine $p_i = (u_i, v_i)$;
- i parametri intrinseci della camera K.

Matematicamente, si cerca di risolvere il sistema non lineare:

$$\lambda_i \cdot K^{-1} \cdot \mathbf{p}_i = R \cdot \mathbf{P}_i + \mathbf{t}, \quad \text{per } i = 1, \ldots, n \tag{2.14}$$

dove λ_i sono le profondità ignote.

Il problema ha teoricamente 6 gradi di libertà ed è risolvibile ponendo n=4, per valori minori come n=3 (P3P) si avrebbero ambiguità di posa, mentre per valori maggiori si avrebbe un sistema sovradeterminato. Ci sono diversi metodi di risoluzione:

- Metodi diretti (P3P)[15]: si calcolano le distanze tra i tre punti, che sono invarianti rispetto alla trasformazione rigida derivando un sistema di equazioni polinomiali nelle profondità λ_i che può essere risolto analiticamente.
- EPnP (Efficient PnP) [33]: usato per n grande permette di rappresentare i punti 3D come combinazione lineare di 4 punti di controllo virtuali. Questo riduce il problema alla stima delle coordinate 3D di questi 4 punti nel sistema camera, trasformandolo in un problema lineare che può essere risolto.
- Metodi iterativi: formulano il problema come ottimizzazione non lineare cercando di minimizzare l'errore di riproiezione.

2.2.1 RANSAC

In presenza di corrispondenze errate, si utilizza il metodo iterativo RANSAC (RANdom SAmple Consensus) [12]. Consiste nel calcolare la posa per un casuale sottoinsieme minimo formato da n elementi e successivamente tramite una soglia sull'errore di riproiezione, conta quanti punti sono inliers con la posa stimata. Il processo si ripete N volte e alla fine si sceglie il modello con il maggior numero di inliers.

2.3 Metriche di valutazione

Le metriche di valutazione possono essere categorizzate in base ai gradi di libertà che valutano, questa sezione esporrà le metriche 6DoF.

2.3.1 ADD - Average Distance of Model Points

La metrica ADD (Average Distance of Model Points) [24] è una delle più utilizzate per valutare la stima della posa di oggetti asimmetrici. Data la rotazione e traslazione ground-truth (R_{gt}, t_{gt}) e quella stimata (R, t), ADD calcola la distanza media tra tutti i punti del modello 3D dell'oggetto trasformati secondo le due pose:

$$\text{ADD} = \frac{1}{|O|} \sum_{x \in O} \|(R_{gt}x + \mathbf{t}_{gt}) - (Rx + \mathbf{t})\| \tag{2.15}$$

dove $|O|$ rappresenta la norma euclidea dell'insieme dei punti del modello CAD dell'oggetto. Una stima della posa è considerata corretta se ADD è inferiore a una soglia, tipicamente il 10% del diametro dell'oggetto (ADD-0.1d).

2.3.2 ADD-S - Average Closest Point Distance

Per gli oggetti simmetrici, la metrica ADD presenta un problema: oggetti con simmetria rotazionale o riflettiva possono avere pose visivamente identiche ma matematicamente diverse. Ad esempio, un cilindro ruotato di 180° attorno al suo asse verticale appare identico, ma i punti corrispondenti del modello 3D si trovano in posizioni diverse. Questo causa valori di ADD elevati anche quando la posa stimata è visivamente corretta.

Per risolvere questa ambiguità intrinseca, è stata introdotta la metrica ADD-S (Average Closest Point Distance) [24], che calcola la distanza media utilizzando il punto più vicino invece della corrispondenza diretta:

$$\text{ADD-S} = \frac{1}{|O|} \sum_{x_1 \in O} \min_{x_2 \in O} \|(R_{gt}x_1 + \mathbf{t}_{gt}) - (Rx_2 + \mathbf{t})\| \tag{2.16}$$

Per ogni punto trasformato secondo la posa ground-truth, ADD-S trova il punto più vicino tra quelli trasformati secondo la posa stimata. Questo rende la metrica insensibile alle simmetrie dell'oggetto, valutando correttamente pose che sono equivalenti da un punto di vista percettivo.

Molti benchmark utilizzano la metrica combinata ADD(S), che applica automaticamente ADD per oggetti asimmetrici e ADD-S per oggetti simmetrici.

2.3.3 AUC - Area Under Curve

Piuttosto che utilizzare una singola soglia, è comune calcolare l'area sotto la curva (AUC) delle metriche ADD o ADD-S valutate a diverse soglie. Questo fornisce una misura più completa delle prestazioni su un range di tolleranze. Ad esempio, l'AUC di ADD-S con soglie fino a 0.1m (indicata come ADD-S AUC < 0.1m) è comunemente utilizzata sul dataset YCB-Video [71].

2.3.4 MSSD - Maximum Symmetric Surface Distance

La metrica MSSD misura la massima deviazione di distanza tra i punti della superficie dell'oggetto nello spazio 3D:

$$e_{MSSD}(\hat{P}, \bar{P}, S_M, V_M) = \min_{S \in S_M} \max_{x \in V_M} \|\hat{P}x - \bar{P}Sx\|_2 \tag{2.17}$$

dove S_M rappresenta l'insieme delle trasformazioni di simmetria globale per l'oggetto e V_M sono i vertici del modello. MSSD gestisce le simmetrie dell'oggetto minimizzando su tutte le trasformazioni simmetriche possibili.

2.3.5 MSPD - Maximum Symmetric Projection Distance

MSPD si concentra sulle discrepanze percepibili escludendo l'allineamento lungo l'asse ottico (Z):

$$e_{MSPD}(\hat{P}, \bar{P}, S_M, V_M) = \min_{S \in S_M} \max_{x \in V_M} \|\text{proj}(\hat{P}x) - \text{proj}(\bar{P}Sx)\|_2 \tag{2.18}$$

dove proj$(\cdot)$ rappresenta la proiezione 2D sul piano immagine. Questa metrica è utile quando l'obiettivo è l'allineamento visuale piuttosto che la precisione geometrica 3D assoluta.

2.3.6 BOP-M - Average Recall

La metrica principale del BOP Challenge (BOP-M) [26] è l'Average Recall (AR) calcolato come media delle tre metriche precedenti (VSD, MSSD, MSPD):

$$\text{BOP-M} = \frac{1}{3}(\text{AR}_{VSD} + \text{AR}_{MSSD} + \text{AR}_{MSPD}) \tag{2.19}$$

Questo approccio bilanciato fornisce una valutazione completa che considera sia la precisione geometrica 3D che l'accuratezza della proiezione 2D, rendendola adatta per confrontare metodi destinati a diverse applicazioni.

2.3.7 n°mcm Metric

Questa metrica quantifica direttamente gli errori di rotazione e traslazione in modo indipendente [58]. Una stima della posa è considerata corretta se l'errore di rotazione è inferiore a n gradi e l'errore di traslazione è inferiore a m centimetri:

$$I_{nmcm}(e_R, e_t) = \begin{cases} 1, & \text{se } e_R < n^\circ \text{ e } e_t < m\text{cm} \\ 0, & \text{altrimenti} \end{cases} \tag{2.20}$$

dove e_R e e_t rappresentano rispettivamente gli errori di rotazione e traslazione. Le configurazioni comuni sono 5°5cm, 5°10cm e 10°10cm. Questa metrica è utilizzata nella stima della posa category-level, dove le dimensioni degli oggetti possono variare all'interno della stessa categoria.

L'errore di rotazione può essere calcolato come la distanza geodetica tra le matrici di rotazione:

$$e_R = \arccos\left(\frac{\text{tr}(R_{gt}^T R) - 1}{2}\right) \cdot \frac{180}{\pi} \tag{2.21}$$

mentre l'errore di traslazione è semplicemente la distanza euclidea:

$$e_t = \|\mathbf{t}_{gt} - \mathbf{t}\| \tag{2.22}$$

Capitolo 3

Materiali e Metodi

Questo capitolo è dedicato alla parte sperimentale del lavoro di ricerca, si concentrerà in primo luogo sull'analisi del generatore di dati sintetici e successivamente verrà analizzato il modello scelto per la predizione di keypoints e vertici della bounding box 3D con successiva stima della posa.

3.1 Generazione dati sintetici

3.1.1 Strumenti chirurgici

Volendo sviluppare un metodo di PE per l'ambito chirurgico gli strumenti analizzati sono gli stessi che possono essere trovati una sala operatoria. La selezione è stata guidata seguendo i criteri di rappresentatività, strumenti utilizzati frequentemente in procedure chirurgiche reali e diversità geometrica, forme diverse per testare la robustezza del metodo. I tre strumenti selezionati sono:

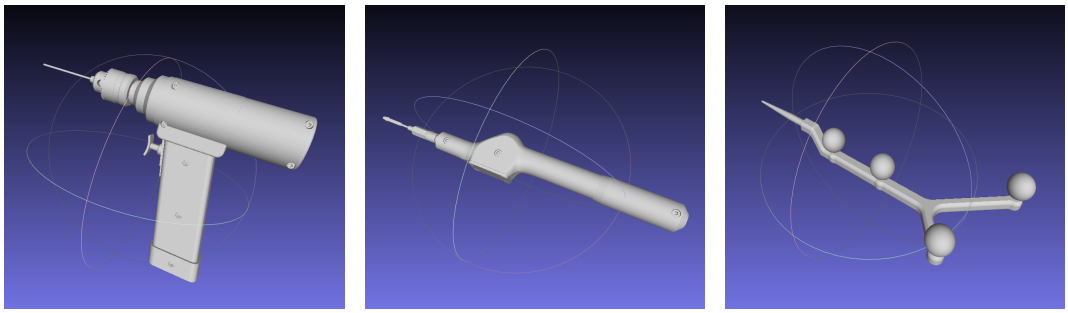
Trapano chirurgico: strumento cilindrico utilizzato per la perforazione ossea, caratterizzato da una struttura cilindrica che nella stima della posa può causare della simmetria rotazionale parziale lungo l'asse longitudinale.

Strumento di taglio per osteotomie: utilizzato per il taglio preciso del tessuto osseo.

Puntatore marker-based: questo strumento è tipicamente utilizzato in sistemi di navigazione chirurgica basati su marker ottici. Include quattro marker sferici posizionati in configurazione tetraedrica per garantire visibilità da diverse angolazioni. L'obiettivo della sua inclusione è valutare se un approccio markerless può raggiungere accuratezza comparabile ai sistemi marker-based tradizionali.

La Fig. 3.1 mostra i modelli 3D degli strumenti visualizzati in MeshLab e realizzati su SolidWorks CAD. Per validare la correttezza dei modelli virtuali e testare il metodo in condizioni reali, gli strumenti sono stati replicati fisicamente utilizzando stampa 3D. La stampa è stata effettuata con usando una stampante Prusa MK3S con un filamento di PLA proveniente dal fornitore TreeD Filament. Per tutti i tool sono stati utilizzati gli stessi parametri: spessore del deposito di 0.07mm, 50% densità di riempimento, 3 perimetri circostanti e motivo di riempimento giroide. Le repliche

fisiche hanno permesso di condurre esperimenti con immagini reali per valutare il *domain gap*.



(a) Trapano chirurgico **(b)** Strumento di taglio **(c)** Puntatore marker-based

Figura 3.1: Modelli 3D dei tre strumenti chirurgici utilizzati.

3.1.2 Annotazioni

Come anticipato nella Sezione 1.5.1, la Perception Camera permette di selezionare una serie di annotazioni da restituire in output.

3.1.2.1 Keypoints

I keypoints rappresentano punti di interesse semanticamente rilevanti sulla superficie dell'oggetto e sono fondamentali per la stima della posa tramite algoritmi PnP. La selezione e il posizionamento dei keypoints deve garantire un'adeguata distribuzione spaziale, coprendo uniformemente l'oggetto per minimizzare l'errore di riproieizione, devono avere una rilevanza semantica, quindi corrisponde ai punti con feature geometriche distintive (e.g. la punta nel trapano) e devono essere robusti alle occlusioni, quindi bisogna evitare configurazioni in cui nessun keypoints è visibile. Per ogni strumento sono stati definiti 8 keypoints, posizionati manualmente in fase di setup tramite l'interfaccia di Unity Perception. La Fig. 3.2 mostra il posizionamento dei keypoints sui tre strumenti. Le coordinate dei keypoints vengono esportate sia in 2D (proiezione sul piano immagine) che in 3D (sistema di riferimento camera), accompagnate da uno stato di visibilità a tre livelli:

- **0 - Occluso/Assente**: il keypoint non è visibile nella camera a causa di occlusioni o perché si trova fuori dal campo visivo;
- **1 - Parzialmente visibile**: il keypoint è parzialmente occluso ma ancora rilevabile;
- **2 - Completamente visibile**: il keypoint è completamente visibile senza occlusioni.

3.1.2.2 Bounding Box

Vengono generate due tipologie:

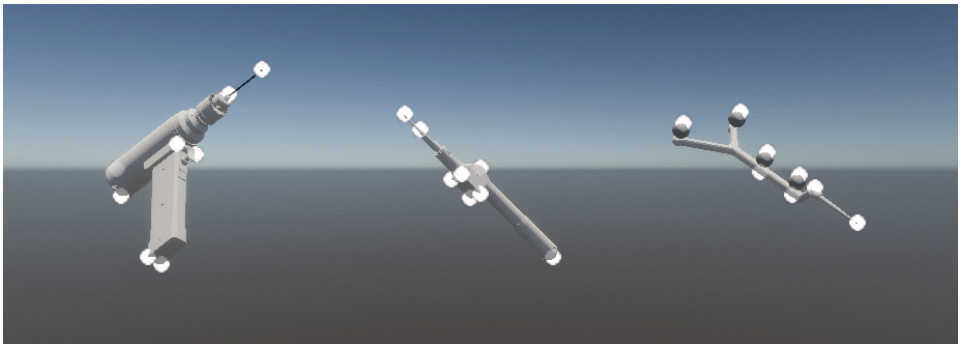


Figura 3.2: Keypoints selezionati su Unity

Bounding box 2D: parallelepipedo allineato agli assi che racchiude strettamente la proiezione dell'oggetto sul piano immagine. È definito da quattro coordinate $(x_{min}, y_{min}, x_{max}, y_{max})$ e viene utilizzato per detection e localizzazione iniziale dell'oggetto e cropping della regione di interesse (ROI) per l'elaborazione successiva.

Bounding box 3D: parallelepipedo orientato nello spazio 3D che racchiude l'oggetto nel suo sistema di riferimento locale. È definito dalle sue dimensioni (w, h, d), larghezza, altezza e profondità, dalla rotazione e dalla traslazione. Gli otto vertici della bounding box 3D sono calcolati nel sistema di riferimento locale come:

$$\mathbf{v}_i^{local} = \begin{bmatrix} \pm\frac{w}{2} \\ \pm\frac{h}{2} \\ \pm\frac{d}{2} \end{bmatrix}, \quad i = 0, 1, \ldots, 7 \tag{3.1}$$

E trasformati nel sistema di riferimento camera applicando la trasformazione rigida:

$$\mathbf{v}_i^{cam} = \mathbf{R} \cdot \mathbf{v}_i^{local} + \mathbf{t} \tag{3.2}$$

3.1.2.3 Maschere di segmentazione

La segmentazione semantica fornisce una maschera binaria pixel-wise che separa l'oggetto dallo sfondo (Fig. 3.3b). Questa annotazione è essenziale per metodi che operano su nuvole di punti o RGBD, dove la maschera permette di isolare i punti appartenenti all'oggetto e per fare post-processing per raffinare le predizioni eliminando outliers esterni alla silhouette dell'oggetto.

La maschera viene generata automaticamente da Unity Perception tramite rendering della scena con shader specializzati che assegnano colori univoci a ogni istanza di oggetto.

3.1.2.4 Mappe di profondità

Le depth map forniscono informazioni metriche sulla distanza di ogni pixel dalla camera (Fig. 3.3c). Vengono salvate in formato EXR a 32-bit floating point per preservare la precisione numerica.

Le depth map sono utilizzate per backprojection di coordinate 2D in coordinate 3D camera tramite:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = Z_c \cdot \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{3.3}$$

dove Z_c è il valore della depth map nel pixel (u, v) e per fare raffinamento della posa sfruttando vincoli geometrici 3D.

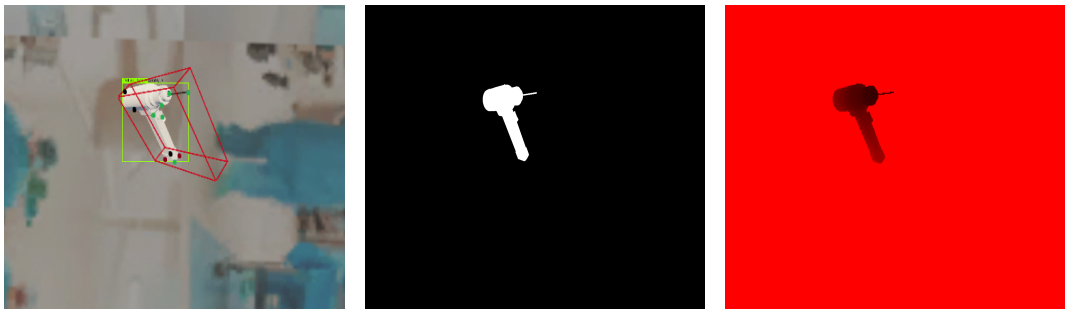
3.1.2.5 Metadati della posa ground-truth

Oltre alle annotazioni visuali, Unity Perception esporta i metadati precisi della posa di ogni oggetto in un file JSON, includendo:

Rotazione: espressa come quaternione unitario $\mathbf{q} = (q_w, q_x, q_y, q_z)$, che viene convertito in matrice di rotazione (Eq. 2.10

Traslazione: vettore 3D $\mathbf{t} = [t_x, t_y, t_z]^T$ che specifica la posizione del centro dell'oggetto nel sistema di riferimento camera.

I file in uscita vengono organizzati secondo il formato SOLO (Synthetic Optimized Labeled Objects), quindi suddivisi in N cartelle, dove N è il numero di frame acquisiti, contenenti la il frame RGB, la segmentazione, la mappa di profondità e un file .json contenente tutte le restanti annotazioni. Però la maggior parte delle reti neurali accetta in input un determinato formato per i file, per questo sono stati generati 5 file di conversione in Python, uno al fine del progetto e i restanti quattro per applicazioni future. I codici creati convertono il formato SOLO per l'addestramento delle reti DenseFusion [71], PVN3D [23], PVNet [48], SingleShotPose [61] e YOLO [50], in aggiunta è stato creato un codice anche per il formato BOP [27].



(a) Acquisizione della camera **(b)** Maschera di segmentazione **(c)** Mappa di profondità

Figura 3.3: Annotazioni in output.
Da sinistra a destra: (a) frame RGB con le bounding boxes 2D (verde) e 3D (rosso) sovrapposte, e in nero i keypoints; (b) maschera di segmentazione, in bianco lo strumento e in nero lo sfondo; (c) mappa di profondità del frame visualizzata su RenderDoc, più il nero è sfumato, maggiore è la distanza del pixel dalla camera.

3.1.3 Sfondi

Una volta definiti gli strumenti da utilizzare si è potuto procedere con la progettazione della pipeline su Unity. L'obiettivo era quello di potere creare scene che richiamassero gli ambienti di utilizzo dei tool. Per questo motivo in Unity è stato inserito un panel che deve fungere da background dinamico, ossia una serie di immagini che cambiassero continuamente da un frame a quello successivo e sono state usate immagini provenienti dal dataset SUN397 [75], creato per allenare modelli per il riconoscimento della scena, e immagini trovate tramite Google Immagini. Sono state selezionate immagini di sale operatorie, stanze di ospedali e reparti industriali fino ad ottenere 500 sfondi differenti. Dopodiché per aggiungere variabilità è stato creato uno script C# che oltre ad occuparsi dell'aggiornamento dell'immagine modificasse la sua orientazione attorno all'asse Y, la dimensione, l'offset di posizionamento lungo l'asse delle x e delle z e randomicamente modificasse luminosità, sfocatura e tonalità; tutto regolato da una serie di parametri definibili dall'utente. In questo modo stessi sfondi su frame differenti sarebbero apparsi differenti. Fig. 3.4 mostra la visuale del randomizzatore su Unity e i parametri scelti in fase di generazione.

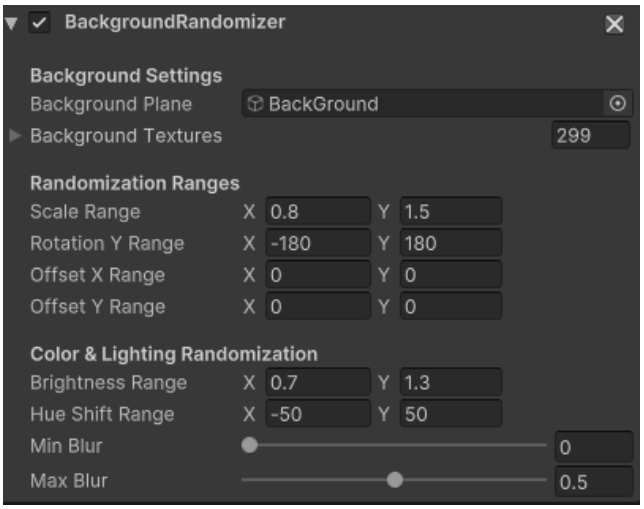


Figura 3.4: Randomizzatore dello sfondo

Un'altra versione dello sfondo è stata usata per creare diverse versioni del dataset, per poi verificare quale fosse la migliore per addestrare il modello. In questa versione non è più una serie di immagini, ma una serie di oggetti disposti casualmente a una determinata distanza dalla camera. Questo serve per creare degli elementi di disturbo sullo sfondo, aiutando la rete a riconoscere ed a escludere elementi che non sono quello da segmentare. Unity Perception fornisce già una scena per generare lo sfondo e utilizza una serie di oggetti di uso comune come bottiglie di plastica, scatole di alimenti e lattine con texture variabili. Un esempio di frame generato è presentato in Fig. 3.5.

Figura 3.5: Sfondo con distrattori

3.1.4 Trasformazioni geometriche

Utilizzando il *Transform Randomizer* integrato in Unity Perception in ogni frame sono state randomizzate casualmente la traslazione, la rotazione e la scala degli strumenti. Volendo creare un dataset più generico possibile sono state usate rotazioni complete attorno ai tre assi, un'altra scelta di progettazione possibile, in particolare per creare un subset utilizzato solo per il testing, è la limitazione delle rotazioni per garantire un range di movimento conforme all'applicazione, quindi immaginando una camera che riprende il campo visivo dell'operatore che tiene in mano lo strumento. la Fig. 3.6 mostra i parametri modificali dall'utente su Unity. Per quanto riguarda la variazione nella scala si è optato per una randomizzazione uniforme su tutti e tre gli assi, altrimenti questo avrebbe portato a deformazioni delle geometrie dello strumento con conseguente distorsione nella posizione originale del keypoints, causando problemi nella stima della posa. In aggiunta è stato creato uno script C# che si occupa di selezionare casualmente uno strumento e posizionarlo nella scena, mantenendo costante il numero di volte che ogni oggetti è apparso, così da evitare scompensi a livello di numerosità delle classi; nel caso questo non fosse l'obiettivo è stato anche creato un paramento modificabile per determinare le percentuali di generazioni di ogni oggetto.

3.1.5 Illuminazione

La scena Unity utilizza un modello di illuminazione basato su:

Directional light: simula una sorgente luminosa distante (come la lampada scialitica principale) con i seguenti parametri randomizzati:

- Intensità: $I_{min} = 5000$ lux e $I_{max} = 80000$ lux
- Temperatura colore: $T_{min} = 2000$ K e $T_{max} = 10000$ K
- Direzione: variando così le ombre nella scena ed è stato usato lo stesso *Transform Randomizer* usato anche per gli strumenti chirurgici.

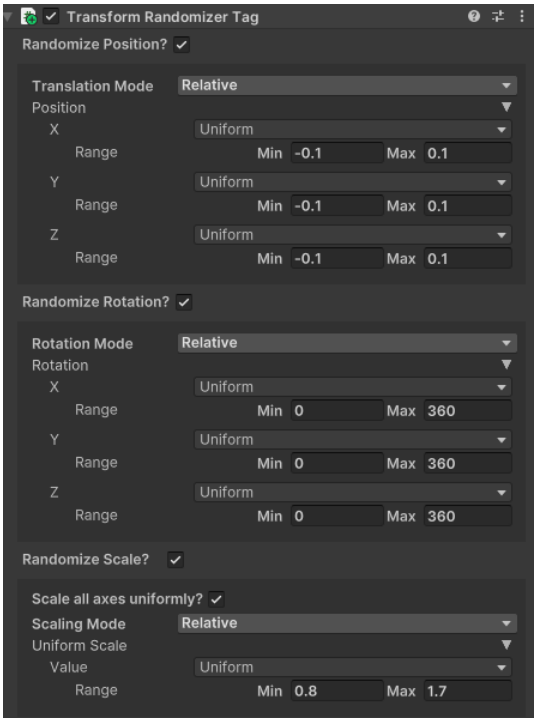


Figura 3.6: Randomizzazione della trasformata

Point Lights: sorgenti luminose puntiformi posizionate casualmente nella scena per simulare fonti secondarie come monitor, apparecchiature elettromedicali luminose, ecc.

3.1.6 Parametri intrinseci

Unity Perception non fornisce direttamente la matrice K (vedi Eq. 3.5), ma solamente la matrice di proiezione. Per determinare i parametri intrinseci della camera è stato creato uno script C# che calcola fx, fy, cx, cy nel seguente modo:

$$\begin{aligned} f_y &= \frac{H/2}{\tan\left(\frac{\text{fov}_y}{2}\right)}, \qquad & f_x &= f_y \cdot \frac{W}{H}, \\ c_x &= \frac{W}{2}, & c_y &= \frac{H}{2}, \end{aligned} \tag{3.4}$$

dove H è l'altezza dell'immagine in pixel, W è la larghezza dell'immagine in pixel, fov_y è il campo visivo verticale espresso in radianti e il rapporto $\frac{W}{H}$ rappresenta l'*aspect ratio* dell'immagine. Per replicare le stesse condizioni del laboratorio, i parametri della camera in Unity sono stati modificati per essere il più simile possibile a quelli della Intel RealSense D435i, la camera usata per le acquisizioni nel reale. In particolare è stata impostata una proiezione prospettica invece che ortogonale, un FOV di 42° sulla verticale, e campo vicino di $0.1m$ e un campo lontano per $> 10m$, una risoluzione di $1280x720$ e un frame rate nominale di 30 FPS. Questo porta ad avere una matrice dei parametri intrinseci pari a:

$$K = \begin{bmatrix} 921 & 0 & 640 \\ 0 & 920 & 360 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

che è stata poi confrontata con i parametri intrinseci della camera calcolati tramite un processo di calibrazione tramite marker ArUco spiegato nella Sez. 3.5. Dopodiché, visto che la rete prende come input immagini $416x224$, questa matrice viene scalata secondo le seguenti formule:

$$\begin{aligned} \text{scale}_x &= \frac{256}{1280} \approx 0.200 \\ \text{scale}_y &= \frac{256}{720} \approx 0.356 \\ f_x^{\text{new}} &= f_x^{\text{original}} \cdot \text{scale}_x \\ f_y^{\text{new}} &= f_y^{\text{original}} \cdot \text{scale}_y \\ c_x^{\text{new}} &= c_x^{\text{original}} \cdot \text{scale}_x \\ c_y^{\text{new}} &= c_y^{\text{original}} \cdot \text{scale}_y \end{aligned}$$

ottenendo così la nuova matrice scalata

$$K_{scaled} = \begin{bmatrix} 184.3 & 0 & 130.07 \\ 0 & 327.5 & 127.66 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

3.1.7 Dataset

Per la fase di addestramento sono state create N versioni differenti del dataset, ognuna caratterizzata dalla variazione controllata di un singolo parametro. Tale approccio ha permesso di condurre un'analisi comparativa sistematica sugli effetti che ciascuna configurazione produce sulle prestazioni del modello. L'obiettivo finale è stato quello di definire un dataset ottimale, che combinasse le caratteristiche più vantaggiose emerse dagli esperimenti condotti sui dataset minori, massimizzando l'accuratezza del modello e al contempo riducendo il costo computazionale dell'addestramento.

I parametri oggetto di variazione e analisi sono i seguenti:

- Numero di frame: è noto che un incremento del numero di campioni nel dataset tenda ad aumentare le metriche di inferenza del modello, poiché ne migliora la capacità di generalizzazione. Tuttavia, la variazione di questo parametro è stata introdotta per individuare un punto di saturazione, ovvero un intervallo di immagini oltre il quale l'aumento di performance risulta marginale rispetto al costo computazionale aggiuntivo. Questo consente di creare dataset più leggeri e rapidi da addestrare, senza compromettere significativamente l'accuratezza complessiva.

- Presenza di frame multi-strumento: la coesistenza di più strumenti nella stessa scena può fornire informazioni contestuali utili al modello. Tale configurazione permette di valutare se la presenza simultanea di più oggetti migliori la capacità discriminativa della rete, specialmente in situazioni di parziale sovrapposizione o occlusione degli strumenti.

- Presenza di distrattori: sono stati inseriti nella scena solidi geometrici di varia forma, colore e texture, con l'obiettivo di testare l'impatto di elementi non rilevanti sulla robustezza e generalizzazione del modello. Questa variazione consente di simulare scenari più realistici, nei quali l'algoritmo deve essere in grado di distinguere con precisione gli strumenti chirurgici dagli elementi di disturbo.

Per garantire una valutazione equa tra i diversi dataset, è stato mantenuto costante il seed di generazione in tutte le versioni. Questa scelta ha consentito di preservare il medesimo posizionamento spaziale e le stesse traiettorie degli strumenti chirurgici, assicurando che le eventuali variazioni nelle prestazioni dei modelli fossero attribuibili esclusivamente alle caratteristiche peculiari introdotte in ciascuna configurazione.

Durante la fase di ricerca delle migliori caratteristiche, il dataset è stato suddiviso in modo che il validation contenesse sempre 100 immagini, tra cui scatti con sfondo monocromatico, sfondo randomizzato, frame multi-strumento e in alcune presenza di distrattori; in questo modo il subset ottenuto è più eterogeneo, permettendo quindi di analizzare le metriche su un ambiente più variabile, come quello reale. In aggiunta mantenendo il contenuto semantico uguale si è escluso che le alterazioni sulle metriche di validazione fossero dovute esclusivamente all'aumentare dei campioni per l'addestramento. Non è stato invece creato un test set, poiché in questa fase le valutazioni delle prestazioni sono state effettuate esclusivamente sul validation set.

Una volta completata l'analisi comparativa e individuati i parametri che rappresentano il miglior compromesso tra accuratezza e efficienza computazionale, verrà generato un dataset finale che integrerà tali caratteristiche ottimali. Quest'ultimo costituirà la base per l'addestramento definitivo dei modelli.

La Tabella 3.1 riassume le principali caratteristiche dei dataset generati, evidenziando le differenze nei parametri di configurazione e le dimensioni complessive di ciascuna versione dividendoli in fasi:

- Fase 1: variazione nel numero di immagini;

- Fase 2: variazione nella percentuale di generazione di frame multi-strumento;

- Fase 3: presenza o meno di distrattori;

- Fase 4: fare tuning degli iper-parametri della rete;

- Fase 5: dataset migliore risultante dalle fasi precedenti con 24000 immagini.

Nella Sez. 3.3 verranno mostrati i risultati per ognuna delle configurazioni del dataset. A seguito delle analisi condotte su quest'ultimi il dataset finale è composto

Tabella 3.1: Statistiche descrittive dei dataset generati per l'addestramento

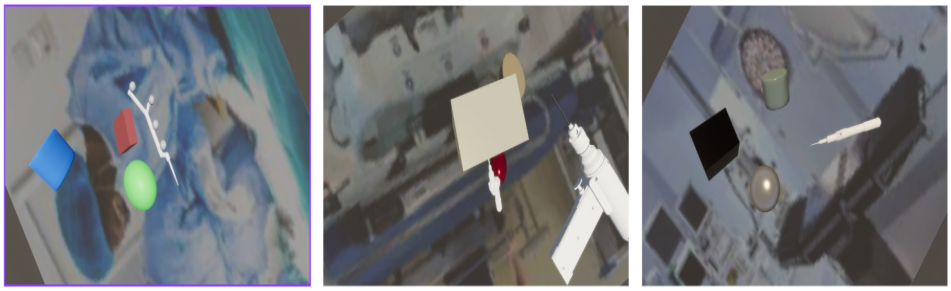
Dataset ID	**Fase**	**Samples**	**% Multi-tool**	**Distrtattori**
A1	1	500	0%	No
A2	1	750	0%	No
A3	1	1000	0%	No
A4	1	1250	0%	No
A5	1	1500	0%	No
A6	1	1750	0%	No
B1	2	1000	0%	No
B2	2	1000	30%	No
B3	2	1000	60%	No
B4	2	1000	90%	No
C1	3	1000	0%	No
C2	3	1000	0%	Yes
A1	4	1000	0%	Yes
A1+	5	24000	0%	Yes

Note:
Samples = numero di immagini complessive
% Multi-tool = percentuale di immagini con più di uno strumento presente
Distrattori = presenza o meno di elementi di disturbo nell'immagine

da 24000 immagini, senza frame multi-strumento e senza distrattori. La suddivisione usata per l'addestramento è stata 70-20-10 %. Analizzando il tempo necessario per la generazione del dataset finale con risoluzione $1280x720$, è stato stimato un tempo medio di generazione pari a 0.055 s per immagine. Come prevedibile, il tempo di processamento varia proporzionalmente alla risoluzione delle immagini.

Alcune immagini di esempio per vedere l'effetto dei randomizzatori esposti fino ad ora sono presentate nella Fig. 3.7.

(a) Immagini con strumenti



(b) Immagini con strumento e distrattori

Figura 3.7: Esempi immagini sintetiche
(a) da sinistra a destra, bassa luminosità, presenza di due strumenti in contemporanea, intensità luminosa media e temperatura calda (b) Immagini con presenza di distrattori colorati a schermo

3.2 PVNet

3.2.1 Overview

Il modello utilizzato per la predizioni dei keypoints e dei vertici delle bounding box 3D è stato PVNet [48], già introdotto in Sez. 1.3.1.

I metodi tradizionali per la stima della posa basati su keypoints presentano limitazioni che ne compromettono l'efficacia in scenari realistici caratterizzati da occlusioni e troncamenti. Quando un keypoint è occluso o si trova al di fuori del campo visivo della camera, la rete fatica a produrre predizioni accurate poiché non ha accesso a informazioni visuali dirette su quel punto specifico.

PVNet supera queste limitazioni sfruttando una proprietà degli oggetti rigidi: date alcune parti visibili di un oggetto, è possibile inferire la direzione relativa verso altre parti, anche se queste sono occluse o fuori dal campo visivo. Questa intuizione si basa sull'osservazione che le relazioni geometriche tra parti di un oggetto rigido sono invarianti rispetto alla posa, quindi vedendo una porzione dell'oggetto possiamo dedurre dove si trovano le altre parti. Invece di predire direttamente la posizione dei keypoints, PVNet predice per ogni pixel appartenente all'oggetto un vettore unitario che punta nella direzione del keypoint target, usando di fatto una rappresentazione a campo vettoriale.

La rete è forzata a concentrarsi su caratteristiche locali dell'oggetto e relazioni spaziali tra parti, piuttosto che su feature globali influenzate dallo sfondo. Ogni pixel

deve apprendere quale direzione puntare basandosi sole sulle sue feature locali e sulla comprensione della struttura geometrica dell'oggetto, rendendo le predizioni meno sensibili al contesto circostante. Anche se un keypoint è completamente occluso, può essere localizzato accuratamente aggregando le predizioni dalle parti visibili dell'oggetto tramite un meccanismo di voting. Se ad esempio la punta di uno strumento chirurgico è occlusa dalla mano dell'operatore, ma il manico è visibile, i pixel del manico possono ancora predire vettori che puntano correttamente verso la punta occlusa, permettendo di inferirne la posizione. I keypoints al di fuori dei confini dell'immagine possono essere rappresentati e localizzati correttamente, poiché i vettori puntano verso posizioni arbitrarie nello spazio 2D. A differenza delle heatmap che sono limitate alle dimensioni dell'immagine, un vettore può puntare in qualsiasi direzione e verso qualsiasi distanza, permettendo di rappresentare keypoints completamente fuori frame. Il processo di voting fornisce una distribuzione di probabilità spaziale per ogni keypoint, che può essere utilizzata per pesare le corrispondenze 2D-3D nell'algoritmo per la risoluzione del problema PnP. Questa informazione di incertezza è preziosa perché keypoints derivanti da una predizione più sicura possono avere peso maggiore nella risoluzione del PnP rispetto a keypoints ambigui o parzialmente occlusi.

3.2.2 Architettura

PVNet adotta un'architettura fully convolutional basata su ResNet-18 [22] come backbone, modificata per produrre predizioni dense pixel-wise. La rete ha due rami di output paralleli che condividono le feature estratte dal backbone. Il primo ramo produce una maschera di segmentazione che identifica i pixel appartenenti a ciascuna classe di oggetto, permettendo di distinguere l'oggetto target dallo sfondo e da altri oggetti presenti nella scena. Il secondo ramo, per ogni keypoint di ogni classe, predice un campo vettoriale che rappresenta la direzione da ogni pixel verso quel keypoint. La condivisione delle feature tra i due rami permette alla rete di apprendere rappresentazioni che sono simultaneamente utili per la segmentazione e per la predizione vettoriale. Fig. 3.8 mostra la struttura della rete.

La rete riceve in input un'immagine RGB di dimensione $H \times W \times 3$, dove H e W sono rispettivamente l'altezza e la larghezza dell'immagine. Non è richiesto alcun preprocessing particolare oltre alla normalizzazione standard delle immagini RGB, rendendo il metodo semplice da integrare in pipeline esistenti. Il tensore di segmentazione in output ha dimensione $H \times W \times (C + 1)$, dove C è il numero di classi di oggetti e il termine aggiuntivo rappresenta la classe background. Per ogni pixel, la rete produce una distribuzione di probabilità sulle $C + 1$ classi tramite una funzione softmax applicata sull'ultima dimensione. Questo permette di identificare con confidenza variabile a quale classe appartiene ogni pixel dell'immagine. Il tensore di campi vettoriali ha dimensione $H \times W \times (K \times 2 \times C)$, dove K è il numero di keypoints per oggetto e il fattore 2 rappresenta le componenti x e y del vettore unitario. Per ogni pixel e per ogni keypoint di ogni classe, la rete predice un vettore 2D che viene successivamente normalizzato a lunghezza unitaria. Questa normalizzazione

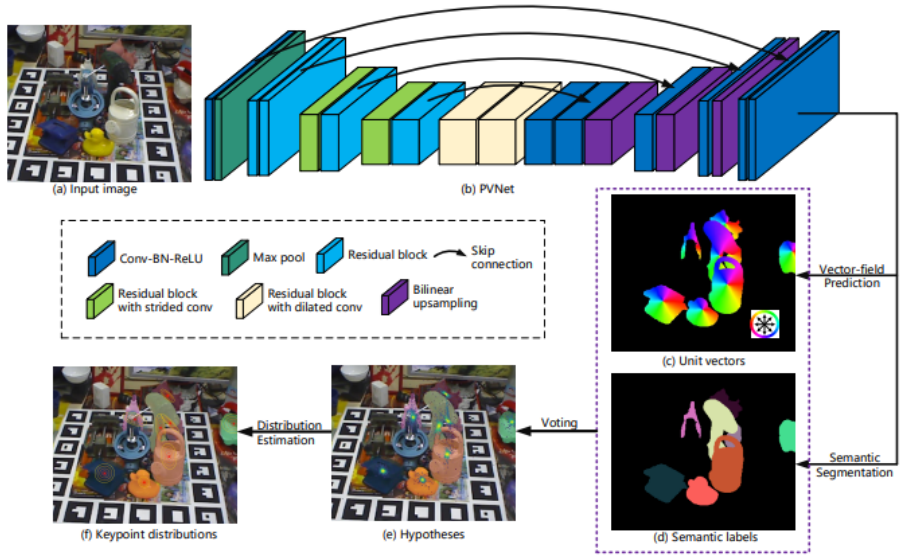


Figura 3.8: Architettura PVNet, immagine proposta nel paper originale [48].

garantisce la proprietà di invarianza di scala.

PVNet introduce tre modifiche principali all'architettura ResNet-18 standard per adattarla alla predizione densa richiesta dal task di stima della posa.

Poiché l'obiettivo è produrre predizioni dense piuttosto che una classificazione globale, i layer finali di global average pooling e fully connected vengono completamente rimossi, mantenendo una struttura fully convolutional end-to-end. Questo permette di preservare l'informazione spaziale attraverso tutta la rete.

Nei residual block finali, le convoluzioni standard vengono sostituite con convoluzioni dilatate (dilated convolutions o atrous convolutions) con dilation rate crescenti. Le convoluzioni dilatate inseriscono 'buchi' (holes) nel kernel convolutivo, permettendo di campionare l'input a una risoluzione spaziale più ampia senza aumentare il numero di parametri. Questo meccanismo permette di aumentare il campo recettivo senza incrementare il numero di parametri, di mantenere una risoluzione spaziale elevata nelle feature map finali evitando eccessivo downsampling, e di catturare contesto a diverse scale spaziali.

Per recuperare la risoluzione spaziale originale dell'immagine, vengono aggiunte skip connections dalle feature map intermedie del backbone verso i layer di output, combinate con operazioni di upsampling bilineare. Questo meccanismo, ispirato all'architettura U-Net [52], permette di preservare dettagli spaziali fini dalle feature map a bassa risoluzione, di combinare informazioni semantiche ad alto livello (che comprendono il contesto globale dell'oggetto) con dettagli geometrici a basso livello (che catturano contorni precisi e texture locali), e di produrre predizioni accurate anche per piccoli oggetti o dettagli fini che altrimenti potrebbero essere persi attraverso i layer di downsampling della rete.

3.2.3 Vettori

PVNet utilizza vettori unitari normalizzati, definiti come:

$$\mathbf{v}_k(p) = \frac{\mathbf{x}_k - p}{\|\mathbf{x}_k - p\|_2} \tag{3.7}$$

dove $\|\cdot\|_2$ denota la norma euclidea. Questa rappresentazione è invariante alla scala poiché dipende solo dalla direzione relativa tra il pixel e il keypoint, non dalla loro distanza assoluta. La normalizzazione rimuove l'informazione sulla magnitudine, mantenendo solo l'informazione direzionale codificata nell'angolo del vettore, permettendo di generalizzare meglio a oggetti a diverse distanze dalla camera, a oggetti con dimensioni fisiche diverse all'interno della stessa categoria, e a variazioni di zoom o risoluzione dell'immagine.

Geometricamente, ogni vettore $\mathbf{v}_k(p)$ può essere interpretato come un raggio che origina dal pixel p e punta verso il keypoint $\mathbf{x}_k$. L'insieme di tutti i vettori per un dato keypoint forma un campo vettoriale che converge radialmente verso la posizione del keypoint, creando una struttura geometrica elegante e interpretabile.

In teoria, l'intersezione di due qualsiasi raggi definiti dai vettori dovrebbe coincidere esattamente con la posizione del keypoint. Infatti, se due pixel p_1 e p_2 predicono perfettamente i vettori $\mathbf{v}_k(p_1)$ e $\mathbf{v}_k(p_2)$, le linee $p_1 + t_1\mathbf{v}_k(p_1)$ e $p_2 + t_2\mathbf{v}_k(p_2)$ si intersecano esattamente in $\mathbf{x}_k$ per opportuni valori di t_1 e t_2.

In pratica, a causa del rumore intrinseco nelle predizioni della rete neurale, dell'approssimazione discreta dei pixel e delle inevitabili imprecisioni nelle annotazioni ground-truth, le intersezioni dei raggi formano una distribuzione di ipotesi attorno alla vera posizione del keypoint piuttosto che convergere esattamente in un punto. Il meccanismo di voting sfrutta questa distribuzione di ipotesi per stimare robustamente la posizione del keypoint e quantificare l'incertezza associata.

3.2.4 Votazione

Una volta generate le ipotesi, ogni pixel dell'oggetto vota per ciascuna ipotesi in base a quanto quella ipotesi è consistente con il vettore predetto dal pixel. Questo meccanismo di voting permette di aggregare l'informazione da tutti i pixel dell'oggetto, non solo dai due pixel utilizzati per generare l'ipotesi, rendendo il processo robusto a predizioni localmente errate. Il voting score $w_{k,i}$ dell'ipotesi $\mathbf{h}_{k,i}$ è definito come una somma pesata di contributi da tutti i pixel dell'oggetto:

$$w_{k,i} = \sum_{p \in \mathcal{P}_{obj}} \exp\left(-\frac{d_{k,i}(p)^2}{2\sigma^2}\right) \tag{3.8}$$

dove $d_{k,i}(p)$ è una misura di distanza che quantifica quanto il pixel p è consistente con l'ipotesi $\mathbf{h}_{k,i}$. Questa distanza è definita geometricamente come la distanza perpendicolare dal pixel p alla linea definita dal vettore $\mathbf{v}_k(p)$ passante per p in direzione dell'ipotesi $\mathbf{h}_{k,i}$:

$$d_{k,i}(p) = \|\mathbf{h}_{k,i} - p\|_2 \cdot \sin(\theta_{k,i}(p)) \tag{3.9}$$

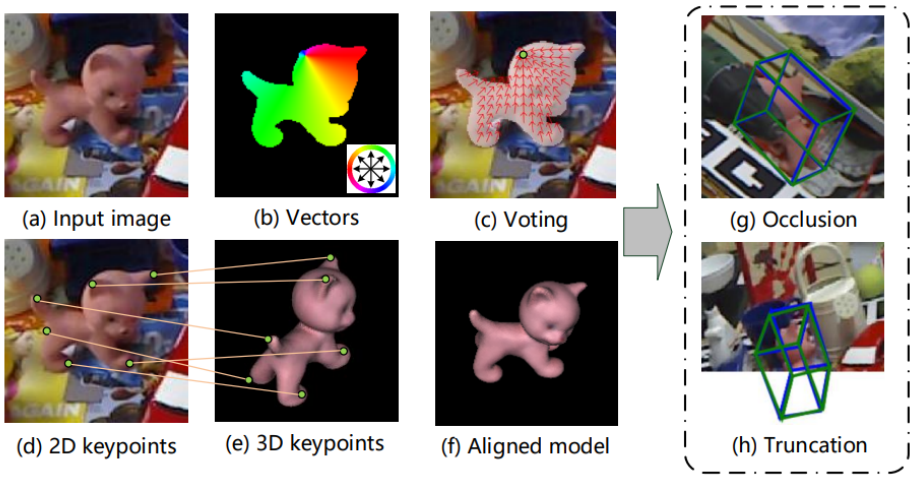


Figura 3.9: Processo di predizione della PVNet, immagine proposta nel paper originale [48]. (a) immagine RGB in input (b) heatmap delle predizioni (c) processo di voting (d,e,f) sequenza per la stima della posa.

dove $\theta_{k,i}(p)$ è l'angolo tra il vettore $(\mathbf{h}_{k,i} - p)$ che va dal pixel all'ipotesi, e il vettore predetto $\mathbf{v}_k(p)$. Quando il vettore predetto punta esattamente verso l'ipotesi, l'angolo è zero e quindi la distanza è zero. Quando il vettore predetto è ortogonale alla direzione verso l'ipotesi, la distanza raggiunge il suo massimo valore pari alla distanza euclidea tra pixel e ipotesi. Un esempio di come la rete predice i keypoints è visibile in Fig. 3.9

3.2.5 Loss di segmentazione

Per il ramo di segmentazione semantica, si utilizza una funzione di cross-entropy pesata calcolata pixel-wise. Data la predizione della rete sotto forma di distribuzione di probabilità sulle classi per ogni pixel e le etichette ground-truth, la loss è definita come:

$$\mathcal{L}_{seg} = -\frac{1}{HW}\sum_{p=1}^{HW} w_p \sum_{c=0}^{C} y_{p,c}\log(\hat{y}_{p,c}) \tag{3.10}$$

dove $y_{p,c}$ è l'indicatore binario che vale 1 se il pixel p appartiene alla classe c e 0 altrimenti (rappresentazione one-hot del ground-truth), $\hat{y}_{p,c}$ è la probabilità predetta dalla rete che il pixel p appartenga alla classe c dopo l'applicazione della softmax, e w_p è un peso assegnato al pixel p per bilanciare il contributo delle diverse classi.

Il peso w_p serve per gestire lo sbilanciamento intrinseco tra pixel oggetto e pixel background. Tipicamente, la maggior parte dei pixel in un'immagine appartiene al background, mentre solo una frazione relativamente piccola appartiene all'oggetto di interesse. Senza pesatura, la rete tenderebbe a minimizzare la loss classificando la maggioranza dei pixel come background, anche a costo di errori sui pixel oggetto. La pesatura può essere implementata assegnando peso maggiore ai pixel delle classi minoritarie, inversamente proporzionale alla loro frequenza nel dataset, oppure usando

schemi più sofisticati come focal loss che penalizza maggiormente gli errori sui pixel difficili da classificare.

3.2.6 Loss del campo vettoriale

Per il ramo di predizione del campo vettoriale, la loss deve incoraggiare la rete a predire vettori che puntino accuratamente verso i keypoints. PVNet propone due formulazioni alternative per questa loss, entrambe valide ed efficaci.

La prima formulazione si basa sulla minimizzazione dell'angolo tra i vettori predetti e ground-truth:

$$\mathcal{L}_{vec} = \frac{1}{K|\mathcal{P}_{obj}|} \sum_{k=1}^{K} \sum_{p \in \mathcal{P}_{obj}} \Big(1 - \mathbf{v}_k(p)^T \hat{\mathbf{v}}_k(p)\Big) \tag{3.11}$$

dove $\mathbf{v}_k(p)$ è il vettore unitario ground-truth calcolato dalla posizione annotata del keypoint k e la posizione del pixel p, $\hat{\mathbf{v}}_k(p)$ è il vettore predetto dalla rete per quel pixel e keypoint (dopo normalizzazione a lunghezza unitaria), e il prodotto scalare $\mathbf{v}_k(p)^T \hat{\mathbf{v}}_k(p)$ calcola il coseno dell'angolo tra i due vettori.

Il termine $1 - \cos(\theta)$ fornisce una misura di dissimilarità che vale zero quando i vettori sono perfettamente allineati (angolo zero, coseno uno) e cresce fino a due per vettori opposti (angolo di 180 gradi, coseno meno uno). Questa loss è geometricamente interpretabile e invariante alla parametrizzazione specifica dei vettori, dipendendo solo dall'angolo relativo.

La seconda formulazione alternativa utilizza una smooth L1 loss applicata direttamente alle componenti dei vettori:

$$\mathcal{L}_{vec} = \frac{1}{K|\mathcal{P}_{obj}|} \sum_{k=1}^{K} \sum_{p \in \mathcal{P}_{obj}} \text{smooth}_{L1}(\mathbf{v}_k(p) - \hat{\mathbf{v}}_k(p)) \tag{3.12}$$

dove la funzione smooth L1, definita come:

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{se } |x| < 1 \\ |x| - 0.5 & \text{altrimenti} \end{cases} \tag{3.13}$$

si comporta come una loss quadratica per errori piccoli (favorendo convergenza rapida vicino al minimo) e come una loss L1 (valore assoluto) per errori grandi (fornendo gradienti costanti che prevengono problemi di gradienti esplosivi). La smooth L1 loss è stata originariamente introdotta nel contesto di object detection ed è nota per essere più robusta a outliers rispetto alla loss L2 pura.

Indipendentemente dalla formulazione scelta, la loss vettoriale è calcolata solo sui pixel che appartengono all'oggetto secondo le annotazioni ground-truth, ovvero $p \in \mathcal{P}_{obj}$. Questo è essenziale perché i vettori per pixel background non sono definiti e non hanno significato. La media è normalizzata per il numero totale di keypoints K e il numero di pixel oggetto $|\mathcal{P}_{obj}|$ per rendere la loss indipendente dalla dimensione dell'oggetto nell'immagine.

3.2.7 Loss totale e bilanciamento

La loss totale utilizzata per l'addestramento end-to-end della rete è una combinazione lineare pesata delle due componenti:

$$\mathcal{L}_{total} = \mathcal{L}_{seg} + \lambda \mathcal{L}_{vec} \tag{3.14}$$

dove λ è un iper-parametro scalare che controlla l'importanza relativa della loss di segmentazione rispetto alla loss vettoriale. La scelta appropriata di λ è importante per garantire che entrambi i task ricevano attenzione adeguata durante l'ottimizzazione.

Se λ è troppo piccolo, la rete potrebbe concentrarsi principalmente sulla segmentazione, trascurando la qualità della predizione vettoriale. Se λ è troppo grande, potrebbe accadere il contrario. Una strategia comune è impostare λ in modo che i gradienti delle due loss abbiano magnitudini comparabili durante le prime fasi del training, garantendo un bilanciamento dinamico tra i due obiettivi.

In pratica, il valore di λ viene tipicamente determinato empiricamente tramite validazione su un set di sviluppo, cercando il valore che massimizza le prestazioni complessive di stima della posa sul task finale. Nel paper originale, un valore di $\lambda = 1$ è risultato efficace per la maggior parte degli oggetti e dataset considerati.

Durante l'addestramento, si utilizza l'ottimizzatore Adam con learning rate iniziale tipicamente nell'ordine di 10^{-4} o 10^{-5}, che viene ridotto progressivamente secondo uno schedule di decadimento (ad esempio dimezzandolo ogni N epoche o quando la loss di validazione smette di migliorare).

3.2.7.1 Uncertainty-driven PnP

I solver PnP tradizionali come EPnP [33] trattano tutte le corrispondenze 2D-3D in modo uniforme, assumendo che tutti i keypoints abbiano la stessa affidabilità. Tuttavia, in scenari reali con occlusioni o troncamenti, diversi keypoints possono avere livelli di confidenza molto diversi:

- Keypoints completamente visibili e non occlusi → alta confidenza
- Keypoints parzialmente occlusi → confidenza media
- Keypoints inferiti da poche parti visibili → bassa confidenza

PVNet sfrutta le distribuzioni probabilistiche stimate dal voting per pesare differenzialmente le corrispondenze nel PnP solver, ottenendo stime di posa più accurate e robuste.

Date le distribuzioni gaussiane $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ per $k = 1, \ldots, K$ keypoints e le loro coordinate 3D $\mathbf{X}_k$ nel sistema di riferimento dell'oggetto, la posa ottimale $(R, \mathbf{t})$ è stimata minimizzando la distanza di Mahalanobis:

$$\min_{R,\mathbf{t}} \sum_{k=1}^{K} (\tilde{\mathbf{x}}_k - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\tilde{\mathbf{x}}_k - \boldsymbol{\mu}_k) \tag{3.15}$$

dove:

$$\tilde{\mathbf{x}}_k = \pi(R\mathbf{X}_k + \mathbf{t}) \tag{3.16}$$

è la proiezione del keypoint 3D secondo la posa candidata, e $\pi(\cdot)$ è la funzione di proiezione prospettica.

La distanza di Mahalanobis $(z-\mu)^T\Sigma^{-1}(z-\mu)$ generalizza la distanza euclidea tenendo conto della struttura di covarianza. Geometricamente, rappresenta una distanza normalizzata rispetto agli assi principali della distribuzione gaussiana:

- Keypoints con piccola incertezza (traccia di $\mathbf{\Sigma}_k$ piccola) contribuiscono maggiormente all'ottimizzazione, poiché $\mathbf{\Sigma}_k^{-1}$ ha autovalori grandi.
- Keypoints con grande incertezza hanno peso minore, permettendo alla soluzione di 'tollerare' maggiori errori di riproiezione su questi punti.
- La matrice di covarianza cattura anche correlazioni direzionali: se un keypoint è localizzabile con precisione lungo una direzione ma non lungo l'altra (es. bordi di oggetti), questo è riflesso nella covarianza anisotropa.

L'Eq. 3.15 è un problema di ottimizzazione non lineare nei minimi quadrati. PVNet lo risolve in primo luogo calcolando una stima iniziale della posa utilizzando EPnP standard su un sottoinsieme di 4 keypoints selezionati in base alla confidenza (quelli con traccia di $\mathbf{\Sigma}_k$ minima). Dopodiché la raffina minimizzando l'Eq. 3.15 usando l'algoritmo di Levenberg-Marquardt, che itera aggiornamenti della forma:

$$\begin{bmatrix} R \\ \mathbf{t} \end{bmatrix}_{n+1} = \begin{bmatrix} R \\ \mathbf{t} \end{bmatrix}_{n} - (J^T W J + \lambda I)^{-1} J^T W \mathbf{r} \tag{3.17}$$

dove J è la Jacobiana dell'errore di riproiezione rispetto ai parametri di posa, W è una matrice diagonale a blocchi contenente le inverse delle covarianze $\mathbf{\Sigma}_k^{-1}$, $\mathbf{r}$ è il vettore di residui, e λ è il parametro di damping di Levenberg-Marquardt.

3.3 Configurazione dell'addestramento e risultati preliminari

L'addestramento dei modelli per la stima dei keypoints e degli otto vertici della bounding box 3D, d'ora in avanti chiamati **KP-PVNet** e **BB-PVNet**, è stato condotto utilizzando come baseline la rete `ResNet18_8s` come backbone, ottimizzata con l'algoritmo di *Adam*. Il tasso di apprendimento iniziale è stato impostato a 10^{-4}, con un decadimento manuale ogni 30 epoche e fattore di riduzione pari a 0.5. Il numero di epoche totali è stato fissato a 50, con una dimensione di batch pari a 4 per il training e 1 per la validazione. La perdita totale è composta dalla somma di due termini principali: la *Cross-Entropy Loss* per la segmentazione e la *Smooth L1 Loss* per la regressione dei vettori di spostamento verso i keypoint, ponderata da un coefficiente di equilibrio pari a 2.0. La procedura di ottimizzazione non prevede regolarizzazione di tipo *weight decay* o *dropout*, mentre per migliorare la robustezza del

modello sono state applicate leggere trasformazioni di data augmentation, includendo rotazioni casuali nel range $[-15°, 15°]$, riflessioni orizzontali con probabilità 0.5 e variazioni cromatiche moderate di luminosità, contrasto e saturazione. La valutazione del modello è stata eseguita ogni 5 epoche, calcolando le metriche di *precision*, *recall*, *F1-score*, *Mean Distance Error* (MDE), distanza di errore media espressa in pixel, e *Percentage of Correct Keypoints* (PCK) impostata allo 5%, che definisce la presenza di un keypoint entro una determinata soglia, calcolata sulla bounding box 2D, rispetto alla posizione orinale.

La Tab. 3.2 riassume i principali iper-parametri adottati per l'addestramento e validazione nella fase 1, 2 e 3.

Tabella 3.2: Principali iperparametri di addestramento del modello PVNet.

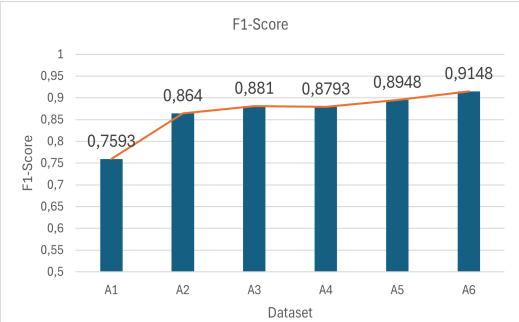
Parametro	**Valore / Descrizione**
Architettura	`ResNet18_8s`
Ottimizzatore	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)
Tasso di apprendimento iniziale	1×10^{-4}
Decadimento del learning rate	Ogni 30 epoche, fattore 0.5
Numero di epoche	50
Batch size (train / val)	8 / 1
Funzione di perdita	Cross-Entropy + Smooth L1 ($\lambda_{\text{vertex}} = 2.0$)
Peso classi (segmentazione)	[1.0, 4.0]
Regularizzazione	Nessuna (weight decay = 0.0, dropout disattivo)
Data augmentation	Rotazioni, flip orizzontale, variazioni cromatiche
Frequenza di validazione	Ogni 5 epoche

Descrizione dei parametri: *Architettura* definisce la rete di base utilizzata per l'estrazione delle feature. *Ottimizzatore* specifica l'algoritmo di aggiornamento dei pesi e i relativi coefficienti di momento. Il *tasso di apprendimento iniziale* controlla l'ampiezza degli aggiornamenti dei pesi ad ogni iterazione, mentre il *decadimento del learning rate* riduce progressivamente tale valore per stabilizzare la convergenza. Il *numero di epoche* indica quante volte l'intero dataset viene elaborato durante l'addestramento. Il *batch size* stabilisce il numero di campioni elaborati simultaneamente. La *funzione di perdita* combina un termine di classificazione (segmentazione binaria) e uno di regressione (predizione dei vettori verso i keypoint). I *pesi di classe* bilanciano l'influenza delle classi maggioritarie e minoritarie nella loss di segmentazione. La sezione di *regolarizzazione* controlla eventuali penalizzazioni sui pesi o dropout, disattivate in questo esperimento. Le tecniche di *data augmentation* migliorano la capacità di generalizzazione del modello variando geometricamente e cromaticamente i dati. Infine, le *metriche di valutazione* permettono di monitorare le prestazioni in termini di accuratezza e coerenza spaziale delle predizioni.

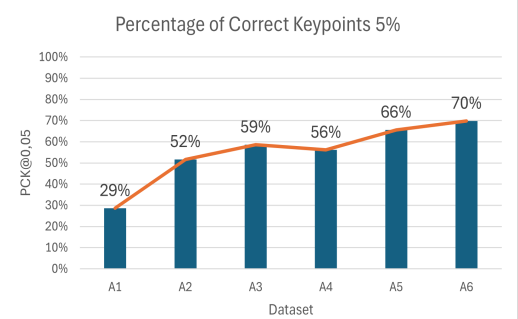
La tabella 3.3 mostra i risultati dei modelli sui diversi dataset. Per quanto riguarda la fase 1, ossia la ricerca di un punto di gomito nelle prestazioni del modello, i risultati posso essere osservati più nel dettaglio nella Fig. 3.10, che mostra gli andamenti del F1-Score e del PCK per i due modelli. I risultati mostrano un trend di miglioramento significativo fino al dataset A3, dopodiché i guadagni in termini di metriche cominciano a crescere in maniera meno impattante. Pertanto per ottenere un compresso iniziale tra efficienza dell'addestramento e precisione dei modelli, le versioni successive dei dataset contengono 1000 immagini.

Analizzando successivamente la fase 2 e la fase 3, dove le configurazioni migliori

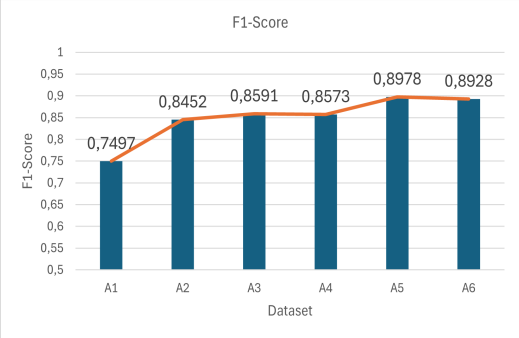
sono quelle che hanno restituito valori di F1-Score più alti e di Mean Distance Error (MDE) più bassi, si può notare una decrescita in termini di performance sia quando nell'immagine sono presenti più strumenti assieme sia quando sono presenti dei distrattori geometrici nell'immagine. La presenza di questi elementi di occlusione potrebbe peggiorare le capacità di apprendimento dei modelli, e il loro utilizzo potrebbe essere più vantaggioso per un futuro fine-tuning della rete allenata su dati grezzi. Una volta definito il miglior dataset su cui allenare KP-PVNet e BB-PVNet, è stato effettuato fine-tuning sugli iper-paramentri (fase 4) al fine di ottenere il risultato più accurato possibile in termini di segmentazione e di predizione dei keypoints. I modelli finali sono stati allenati su un dataset sintetico composto da 24000 immagini, con le caratteristiche migliori scelte tra quelle emerso dallo studio precedente.



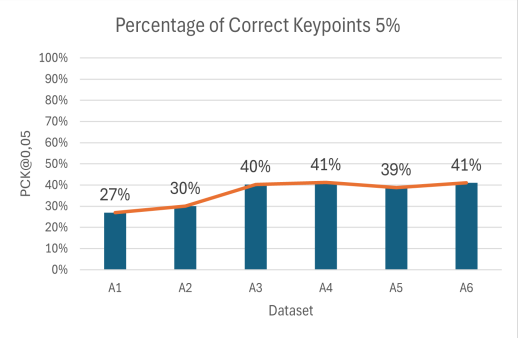
(a) F1-Score KP-PVNet



(b) PCK@0.05 KP-PVNet



(c) F1-Score BB-PVNet



(d) PCK@0.05 BB-PVNet

Figura 3.10: Risultati preliminari della fase 1

Una volta individuato il dataset ottimale, costituito da 1000 immagini, privo di frame multi-strumento e di distrattori (A1), è stata avviata la fase di fine tuning degli iper-parametri dei modelli su tale configurazione. Per completezza e chiarezza espositiva, tutti gli iper-parametri esplorati sono riportati nella Tab. 3.4, evidenziando in grassetto quelli selezionati per l'addestramento finale.

Infine, è stato condotto l'addestramento finale dei modelli sul dataset A1+, con gli iper-parametri selezionati dopo la fase 4. I risultati sul validation set vengono mostrati in Tab. 3.5.

Tabella 3.3: Risultati sperimentali delle configurazioni di rete sul validation set — Fasi 1–3.

ID Modello	Tipo	Dataset	F1-Score	MDE (px)	PCK@0.05
Fase 1: Numero di immagini					
M1.1	KP	A1	0.7593 ± 0.1839	25.85 ± 24.81	29%
	BB	A1	0.7497 ± 0.1622	32.33 ± 12.66	27%
M1.2	KP	A2	0.8640 ± 0.0686	15.44 ± 5.99	52%
	BB	A2	0.8452 ± 0.0838	26.55 ± 14.33	30%
M1.3	KP	A3	0.8810 ± 0.0701	13.85 ± 5.69	59%
	BB	A3	0.8591 ± 0.0919	19.81 ± 9.51	40%
M1.4	KP	A4	0.8793 ± 0.0750	14.53 ± 6.38	56%
	BB	A4	0.8573 ± 0.1024	21.47 ± 15.95	41%
M1.5	KP	A5	0.8948 ± 0.0626	12.56 ± 4.68	66%
	BB	A5	0.8978 ± 0.0589	18.99 ± 6.19	39%
M1.6	KP	A6	0.9148 ± 0.0504	11.96 ± 5.37	70%
	BB	A6	$0,8928 \pm 0.0752$	19.51 ± 7.76	41%
Fase 2: Percentuale di frame multi strumento					
M2.1	KP	A1	0.8810 ± 0.0701	13.85 ± 5.69	59%
	BB	A1	0.8591 ± 0.0919	19.81 ± 9.51	40%
M2.2	KP	B2	0.8301 ± 0.1616	15.94 ± 12.66	27%
	BB	B2	0.7597 ± 0.1622	32.33 ± 12.66	27%
M2.2	KP	B2	0.8441 ± 0.0882	17.98 ± 7.22	56%
	BB	B2	0.7923 ± 0.0965	23.11 ± 9.87	38%
M2.3	KP	B3	0.8057 ± 0.1031	22.37 ± 8.55	49%
	BB	B3	0.7810 ± 0.1188	27.89 ± 10.44	33%
Fase 3: Presenza di distrattori					
M3.1	KP	A1	0.8810 ± 0.0701	13.85 ± 5.69	59%
	BB	A1	0.8591 ± 0.0919	19.81 ± 9.51	40%
M3.2	KP	C2	0.8428 ± 0.0974	19.85 ± 7.44	53%
	BB	C2	0.8213 ± 0.1068	25.16 ± 9.23	36%

Tabella 3.4: Fase 4: fine tuning iper-parametri della rete.

Iperparametro	**Valori testati**
Learning rate (lr)	`[1e-4, 5e-5, 1e-5]`
Optimizer	`[Adam, AdamW, SGD (momentum=0.9)]`
Weight decay	`[0, 1e-5, 1e-4, 5e-4]`
Batch size	`[8, 16, 32]`
Peso classi	`[1-1, 1-2, 1-4]`
Augmentation strength	`[low, medium, high]`

Augmentation strength, si riferisce ai valori di rotazioni delle immagini, alle percentuali di flip e all'intensità delle variazioni cromatiche.

Tabella 3.5: Fase 5: risultati sperimentali delle configurazioni di rete sul validation set.

ID Modello	**Tipo**	**Dataset**	**F1-Score**	**MDE (px)**	**PCK@0.05**
M5.1	KP	A1+	**0.9175 ± 0.0932**	**7.17 ± 4.79**	**95.7%**
M5.1	BB	A1+	**0.8840 ± 0.1792**	**9.46 ± 4.83**	**88.1%**

MDE = Mean Distance Error (in pixel, lower is better).
PCK@0.05 = Percentage of Correct Keypoints entro il 5% della bounding box.
KP = rete per la predizione dei keypoint dello strumento.
BB = rete per la predizione dei vertici della bounding box 3D.

3.4 Validazione nel Reale

Come illustrato nella parte introduttiva di questo lavoro, il *domain gap* è la difficoltà della rete nel passare da analizzare gli elementi del dominio sintetico al dominio reale. Per valutare questo distacco il modello allenato sul dataset artificale è stato testato con condizioni reali in laboratorio. Il problema principale è stato trovare un modo per poter posizionare gli strumenti nello spazio reale e sapere in quale posa si trovassero. Invece di ricorrere ai metodi esposti in Sez. 1.4, è stato ideato un supporto in materiale plastico inserito su una rotaia metallica che permettesse di posizionare gli strumenti in un orientamento conosciuto.

3.4.1 Setup Sperimentale

Per validare le prestazioni del modello di predizione dei keypoints in condizioni reali, è stato progettato e realizzato un sistema di testing controllato che permette di acquisire un ground truth accurato delle pose degli strumenti chirurgici.

Il setup è basato su un struttura formata da rotaie in metallo e supporti specifici stampati con una stampante 3D. La camera è posizionata in modo che possa scorrere sia orizzontalmente che verticalmente per potere variare le condizioni di acquisizione, mentre gli strumenti sono posizionati su un supporto che si può allontanare o avvicinare alla camera. A sorreggere i tool c'è un supporto a base ottagonale, questo permette rotazioni a passi di 45° lungo l'asse passante per il centro del supporto; ne sono state stampate due versioni, una dritta e una inclinata sull'azimut di 45°, permettendo posizionamenti obliqui. Sono state testate anche versioni del supporti

con base dodecagonale, ma queste sono state scartate a causa di un elevato slittamento del supporto attorno all'inserto, rendendo più inaccurati i posizionamenti. Questa configurazione permette di generare in totale 24 rotazioni differenti per ogni strumento. La Fig. 3.11 mostra il supporto montato in laboratorio.

(a) Struttura del supporto, strumenti analizzati, attacco dritto e inclinato.

(b) Struttura per il posizionamento della camera.

Figura 3.11: Struttura del supporto per il posizionamento degli strumenti chirurgici in pose note.

Il supporto creato permette quindi di posizionare lo strumento in una determinata posa utilizzabile come GT in fase di verifica. Ma la rotazione e la traslazione non sono sufficienti, sono necessarie anche la maschera binaria per le metriche di segmentazione e le posizioni dei keypoints 2D e 3D per le metriche di predizione della PVNet.

La maschera binaria è ottenuta tramite *Segment Anything Model* (SAM) [32], un modello di segmentazione universale sviluppato da Meta AI che è in grado di segmentare qualsiasi oggetto in un'immagine mediante prompt interattivi, senza necessità di addestramento specifico per la classe di oggetti target. SAM è composto da tre componenti principali: un *image encoder* basato su vision transformer (ViT) [9] che elabora l'immagine di input una sola volta estraendo feature rappresentative ad alta dimensionalità, un *prompt encoder* che codifica i suggerimenti dell'utente come click positivi (punti appartenenti all'oggetto) o click negativi (punti appartenenti allo sfondo), e un *mask decoder* che combina le feature dell'immagine con i prompt codificati per generare maschere di segmentazione. Il decoder produce tre maschere con diversi livelli di granularità, ciascuna associata a uno score di confidenza, permettendo la selezione automatica della migliore o il raffinamento iterativo mediante ulteriori click. Nel workflow di annotazione, l'immagine RGB acquisita dalla RealSense D435i viene caricata in SAM e l'operatore fornisce pochi click sullo strumento chirurgico per ottenere una maschera accurata in pochi secondi, riducendo drasticamente i tempi di annotazione manuale rispetto ai metodi tradizionali. La maschera così generata è stata successivamente elaborata con una pipeline di post-processing per la rimozione di zone con pochi pixel rilevate come oggetto e il riempimento di buchi all'interno della maschera tramite operatori morfologici di apertura e chiusura.

I keypoints 2D e 3D necessari per le metriche di predizione della PVNet sono ottenuti mediante proiezione prospettica delle coordinate 3D note dal modello CAD dello strumento. Dato che le coordinate dei keypoints nel modello CAD sono espresse rispetto al pivot dello strumento e che la rete richiede coordinate ricentrate rispetto al centro della bounding box 3D, è necessario applicare una traslazione iniziale sottraendo il vettore del centro della bounding box a ciascun keypoint. Successivamente, viene applicata una matrice di conversione del sistema di coordinate da Unity (left-handed con asse Y verso l'alto) al sistema di coordinate della camera RealSense (right-handed con asse Y verso il basso), invertendo la componente Y di ciascun keypoint. Per ottenere i keypoints 2D nell'immagine, si utilizza il modello pinhole con i parametri intrinseci della camera (lunghezze focali f_x, f_y e centro ottico c_x, c_y) estratti direttamente dalla RealSense D435i e verificati tramite calibrazione. La posizione 3D del centro dello strumento rispetto alla camera viene determinata interattivamente: l'operatore posiziona il cursore sul centro della bounding box dello strumento nell'immagine e, mediante un click, il sistema calcola la posizione 3D utilizzando una funzione apposita che combina le coordinate pixel con la profondità misurata dal sensore depth. I keypoints 3D in coordinate locali dello strumento vengono quindi trasformati in coordinate camera mediante la matrice di roto-traslazione $[R|t]$ dove R è la matrice di rotazione (matrice identità per lo strumento in posa neutra) e t è il vettore di traslazione corrispondente alla posizione 3D del centro dello strumento. Infine, i keypoints in coordinate camera vengono proiettati sul piano immagine mediante la matrice dei parametri intrinseci K, ottenendo le coordinate 2D normalizzando per la componente di profondità. Per compensare eventuali discrepanze di scala tra il modello CAD e le dimensioni reali dello strumento, il sistema prevede un fattore di scala regolabile interattivamente dall'operatore mediante i tasti '+' e '-', permettendo di raffinare l'allineamento dei keypoints proiettati con le posizioni reali visibili nell'immagine prima del salvataggio delle annotazioni in formato JSON. La Fig. 3.12 mostra cosa vede l'utente durante il processo di selezionamento dei keypoints del trapano chirurgico

3.5 Calibrazione della camera

La calibrazione della camera è un processo che consente di determinare i parametri intrinseci ed estrinseci della camera, necessari per correggere le distorsioni ottiche e per stabilire una corrispondenza accurata tra le coordinate bidimensionali dell'immagine e le coordinate tridimensionali della scena reale.

Il processo di calibrazione mira a determinare due categorie di parametri che caratterizzano completamente il sistema di acquisizione. I parametri intrinseci descrivono le caratteristiche interne della camera e comprendono la lunghezza focale che rappresenta l'intersezione dell'asse ottico con il piano dell'immagine, e il coefficiente di inclinazione γ che descrive l'eventuale non ortogonalità degli assi dell'immagine; questi parametri sono racchiusi nella matrice intrinseca K, già definita nella Eq. 2.5. I parametri estrinseci, invece, descrivono la posizione e l'orientamento della camera

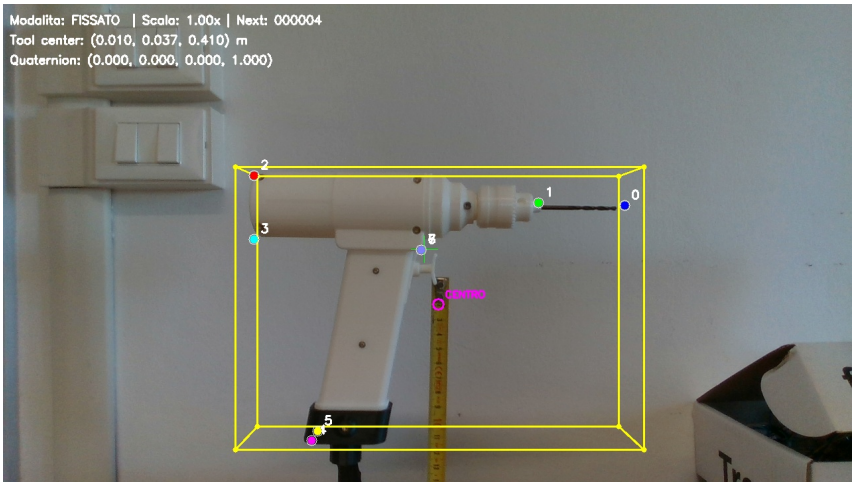


Figura 3.12: Esempio di output dal processo di annotazione manuale. I punti colorati rappresentano i keypoints dello strumento, il parallelepipedo giallo è la bounding box 3D, in viola viene segnato il centro del CAD, in alto a sinistra si ricavano le informazioni di traslazione e rotazione.

rispetto al sistema di coordinate del mondo reale, e sono rappresentati dalla matrice di rotazione R dal vettore di traslazione $\mathbf{t}$. Oltre a questi parametri, è necessario modellare la distorsione delle lenti, che si manifesta principalmente come distorsione radiale, descritta dai coefficienti k1 e k2, e talvolta da distorsione tangenziale, caratterizzata dai coefficienti p1 e p2.

Il metodo proposto da Zhang [80] rappresenta una delle tecniche più diffuse e affidabili per la calibrazione della camera. Questo approccio si distingue per la sua flessibilità ed efficacia, richiedendo solamente un pattern planare a scacchiera osservato da diverse angolazioni. La tecnica si basa sul concetto di omografia, una trasformazione proiettiva planare che descrive la relazione tra i punti tridimensionali sul piano della scacchiera e le loro proiezioni bidimensionali nell'immagine. Il pattern a scacchiera viene utilizzato perché presenta una struttura geometrica regolare e facilmente riconoscibile, caratterizzata da una griglia di quadrati alternati bianchi e neri. In questo lavoro, si utilizza una scacchiera artigianale con marker ArUco, che combina i vantaggi della scacchiera tradizionale con la robustezza dei marker fiduciari per il rilevamento automatico. Gli angoli interni della griglia, comunemente chiamati corner, possono essere rilevati automaticamente con elevata precisione mediante algoritmi dedicati, mentre i marker ArUco forniscono punti di riferimento univoci che facilitano l'identificazione della posa del pattern anche in condizioni di illuminazione variabile o occlusioni parziali. Gli angoli interni di questa griglia, comunemente chiamati corner, possono essere rilevati automaticamente con elevata precisione mediante algoritmi dedicati. Per convenzione, si assume che il piano della scacchiera corrisponda al piano z=0 del sistema di coordinate del mondo, semplificando così la formulazione matematica del problema.

Data un'immagine del pattern di calibrazione, è possibile stimare la matrice di omografia H che mette in relazione i punti del piano della scacchiera con i loro corrispondenti nell'immagine. Questa matrice di dimensione 3×3, definita a meno di

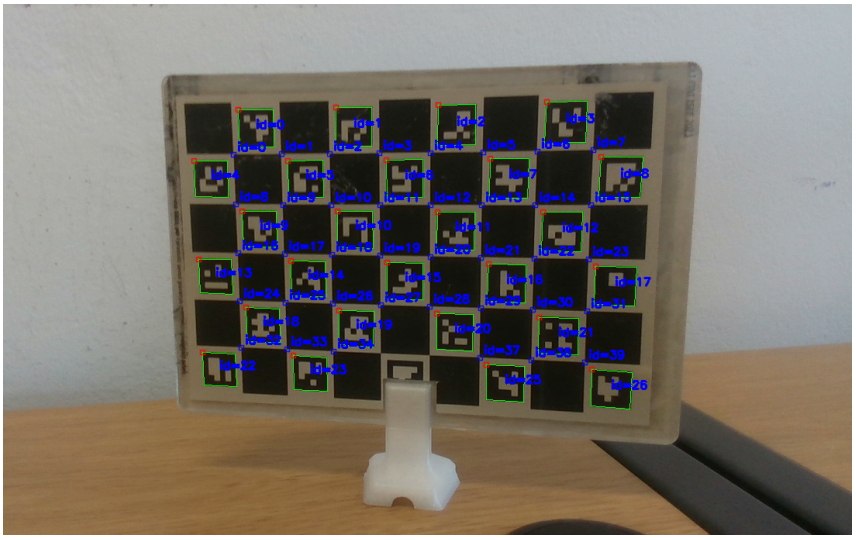


Figura 3.13: Scacchiera ArUco

un fattore di scala, può essere espressa in termini dei parametri intrinseci ed estrinseci della camera. Ogni immagine della scacchiera fornisce due vincoli indipendenti sui parametri intrinseci. Pertanto, per stimare completamente i cinque parametri intrinseci incogniti contenuti nella matrice K, sono necessarie almeno tre immagini del pattern acquisite da diverse orientazioni.

Il processo di calibrazione si articola in diverse fasi. Inizialmente, si acquisiscono multiple immagini della scacchiera da diverse pose, avendo cura che il pattern sia completamente visibile e che le orientazioni relative siano sufficientemente diverse tra loro per garantire la robustezza della stima. È importante evitare configurazioni degeneri, come ad esempio immagini nelle quali il piano della scacchiera subisce una pura traslazione parallela al piano dell'immagine, poiché tali configurazioni non apportano vincoli aggiuntivi utili alla calibrazione. Successivamente, per ciascuna immagine acquisita, si procede al rilevamento automatico dei corner della scacchiera, questo processo viene mostrato in Fig. 3.13. Una volta individuati i corner nell'immagine e note le loro coordinate nel sistema di riferimento del mondo (che possono essere definite in base alle dimensioni note dei quadrati della scacchiera), si calcolano le matrici di omografia per ciascuna vista mediante tecniche come la *Direct Linear Transformation*. Le omografie così ottenute vengono utilizzate per formulare un sistema di equazioni lineari che permettono di calcolare una soluzione iniziale in forma chiusa per i parametri intrinseci. Questo avviene attraverso la determinazione di una matrice B, legata alla matrice intrinseca attraverso una relazione algebrica che può essere risolta mediante decomposizione di Cholesky. Una volta stimati i parametri intrinseci iniziali, è possibile ricavare i parametri estrinseci per ciascuna vista utilizzando la relazione tra l'omografia e le matrici $[R|\mathbf{t}]$.

La soluzione in forma chiusa fornisce una stima iniziale che, sebbene già ragionevolmente accurata, non tiene conto della distorsione delle lenti né degli errori di misura presenti nei dati. Per questo motivo, la fase finale del processo di calibrazione consiste in un'ottimizzazione non lineare che raffina tutti i parametri simultanea-

mente. L'obiettivo di questa ottimizzazione è minimizzare l'errore di riproiezione, definito come la somma delle distanze al quadrato tra i corner osservati nell'immagine e i corner riproiettati utilizzando i parametri stimati. Tale ottimizzazione viene tipicamente condotta mediante l'algoritmo di Levenberg-Marquardt, che combina i vantaggi del metodo del gradiente e del metodo di Gauss-Newton. Durante questa fase di raffinamento, vengono inclusi anche i coefficienti di distorsione radiale e tangenziale nel modello. La distorsione radiale modifica la posizione dei punti dell'immagine in funzione della loro distanza dal centro ottico secondo una relazione non lineare che coinvolge i coefficienti k1 e k2. Questo tipo di distorsione è particolarmente rilevante nelle camere dotate di lenti grandangolari. La distorsione tangenziale, invece, è causata da eventuali disallineamenti tra le lenti e modifica la posizione dei punti secondo direzioni tangenziali rispetto al centro ottico, ed è descritta dai coefficienti p1 e p2.

Una volta completata la procedura di calibrazione, è importante valutare la qualità dei risultati ottenuti. L'errore di riproiezione medio fornisce una misura quantitativa della precisione della calibrazione: valori tipicamente accettabili sono nell'ordine di una frazione di pixel. Inoltre, è possibile verificare visualmente la qualità della calibrazione confrontando i corner osservati con quelli riproiettati utilizzando i parametri stimati oppure applicando la correzione della distorsione a immagini di test contenenti linee rette, che dovrebbero apparire effettivamente rettilinee dopo la correzione.

3.5.1 Fine-tuning su dati reali

Nonostante l'utilizzo di dati sintetici generati con Unity Perception offra numerosi vantaggi in termini di scalabilità e annotazione automatica, esiste inevitabilmente un *domain gap* tra le immagini sintetiche e quelle acquisite in scenari reali. Questo divario si manifesta in differenze di illuminazione, texture dei materiali, rumore del sensore, riflessioni speculariche e altre caratteristiche fotometriche difficili da replicare perfettamente in ambiente virtuale. Per mitigare questo problema e migliorare la robustezza del modello in condizioni operative reali, è stata adottata una strategia di *fine-tuning* sui dati reali.

Il processo di fine-tuning consiste nel continuare l'addestramento del modello PVNet, precedentemente allenato su dataset completamente sintetici, utilizzando un numero limitato di immagini reali acquisite con la camera RealSense D435i. Questa tecnica di *transfer learning* permette al modello di adattare i propri pesi alla distribuzione dei dati reali, mantenendo al contempo le rappresentazioni di alto livello apprese durante il training sintetico.

Per evitare overfitting sul ridotto dataset reale (composto da circa 300 immagini), sono state adottate diverse strategie di regolarizzazione: learning rate differenziati tra backbone ($lr = 5 \times 10^{-7}$) e head ($lr = 5 \times 10^{-5}$) per preservare le feature estratte dal backbone ResNet-18; weight decay ($\lambda = 10^{-4}$) per penalizzare pesi di grande magnitudine; data augmentation aggressiva durante il training (rotazioni fino a $\pm 30°$,

scaling nel range [0.8, 1.2], variazioni fotometriche); early stopping con patience di 20 epoche per interrompere l'addestramento al primo segno di overfitting sul validation set. Il fine-tuning è stato condotto per un massimo di 100 epoche, con split train/validation 80/20, monitorando costantemente le metriche di segmentazione e l'errore medio di localizzazione dei keypoints in pixel.

Questa strategia ibrida iniziale su larga scala con dati sintetici seguito da fine-tuning su dati reali limitati ha dimostrato di essere particolarmente efficace nel bilanciare la necessità di grandi quantità di dati annotati con l'adattamento al dominio target, riducendo significativamente il domain gap senza richiedere annotazioni manuali su migliaia di immagini reali. Infatti modelli allenati puramente sui dati sintetici e testati sui dati reali hanno restituito metriche insufficienti, con F1-score inferiori allo 0.15, segno di evidente overfitting al dominio sintetico.

3.5.2 Metriche di valutazione

Le prestazioni del modello sono quantificate attraverso due metriche complementari per la stima delle predizioni.

Errore Euclideo Medio (Mean Euclidean Distance) La metrica primaria è la distanza euclidea media in pixel tra keypoints predetti e ground truth:

$$\text{MED} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(\hat{u}_i - u_i)^2 + (\hat{v}_i - v_i)^2} \tag{3.18}$$

dove N è il numero totale di keypoints. Questa metrica fornisce una misura intuitiva dell'accuratezza spaziale del modello.

Percentage of Correct Keypoints (PCK) Per valutare la percentuale di predizioni entro una soglia di errore accettabile, si utilizza la metrica PCK, che partendo la distanza euclidea tra keypoint predetto e ground truth

$$d_i = \|\hat{\mathbf{k}}_i - \mathbf{k}_i\|_2 = \sqrt{(\hat{x}_i - x_i)^2 + (\hat{y}_i - y_i)^2} \tag{3.19}$$

definisce una soglia basata sulla grandezza della bounding box

$$\tau = \alpha \cdot S_{\text{bbox}} \tag{3.20}$$

e definisce il numero di keypoints corretti e visibili

$$c_i = \begin{cases} 1, & \text{se } d_i < \tau \\ 0, & \text{altrimenti} \end{cases} \tag{3.21}$$

$$v_i = \begin{cases} 1, & \text{se } \mathbf{k}_i \neq (0,0) \\ 0, & \text{altrimenti} \end{cases} \tag{3.22}$$

$$\text{PCK@0.05} = \frac{\sum_{i=1}^{N} c_i\, v_i}{\sum_{i=1}^{N} v_i} \tag{3.23}$$

dove α è la soglia sotto cui un keypoint è considerato posizionat correttamente, tipicamente si hanno valori di α di 5%, 10%, 20%.

3.6 Ablazione

3.6.1 CycleGAN

Durante lo sviluppo della pipeline di generazione dati, è stata esplorata l'applicazione di CycleGAN per il trasferimento automatico di dominio da immagini sintetiche a reali. L'obiettivo era ridurre il domain gap tra i render Unity degli strumenti chirurgici e le fotografie reali.

CycleGAN è un'architettura per il trasferimento di dominio *unpaired*, cioè che non richiede coppie di immagini corrispondenti tra i due domini. Il framework utilizza due generatori G_A e G_B per le traduzioni bidirezionali tra dominio A (sintetico) e dominio B (reale), insieme a due discriminatori D_A e D_B che distinguono le immagini reali da quelle generate.

La funzione obiettivo combina tre componenti principali. La loss avversaria standard forza i generatori a produrre immagini indistinguibili dalle immagini target:

$$\mathcal{L}_{GAN}(G, D, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D(G(x)))] \quad (3.24)$$

La *cycle consistency loss* garantisce che la traduzione sia invertibile, preservando il contenuto dell'immagine originale:

$$\mathcal{L}_{cyc}(G_A, G_B) = \mathbb{E}_{a \sim A}[\|G_B(G_A(a)) - a\|_1] + \mathbb{E}_{b \sim B}[\|G_A(G_B(b)) - b\|_1] \quad (3.25)$$

L'*identity loss* opzionale preserva i colori quando l'input appartiene già al dominio target:

$$\mathcal{L}_{idt}(G_A, G_B) = \mathbb{E}_{b \sim B}[\|G_A(b) - b\|_1] + \mathbb{E}_{a \sim A}[\|G_B(a) - a\|_1] \quad (3.26)$$

La loss finale è quindi la somma pesata di questi tre contributi.

Il training è stato condotto utilizzando l'implementazione PyTorch di CycleGAN con le seguenti configurazioni. Il dataset consisteva di 500 immagini sintetiche (dominio A) generate con Unity, rappresentanti render fotorealistici di tool industriali su sfondi variabili, e 150 fotografie reali (dominio B) acquisite con iPhone XR. Le immagini sintetiche avevano risoluzione 1280×720 pixel, mentre le fotografie reali variavano fino a 4032×3024 pixel.

L'architettura utilizzata prevedeva generatori basati su ResNet con 9 blocchi residuali e discriminatori PatchGAN con 3 layer convoluzionali. Il preprocessing applicava resize a 286×286 pixel seguito da crop casuale a 256×256 per data augmentation. I parametri di training includevano learning rate iniziale di 0.0002, batch size unitario, e ottimizzatore Adam con $\beta_1 = 0.5$. Il training si è svolto per 200 epoch totali, con learning rate costante per i primi 100 epoch e linear decay per i successivi 100. I pesi delle loss erano $\lambda_{cyc} = 10.0$ per entrambe le direzioni e $\lambda_{idt} = 0.5$.

L'evoluzione delle loss durante il training rivela diverse problematiche critiche. La Tabella 3.6 riporta i valori medi delle loss a intervalli regolari.

Tabella 3.6: Evoluzione delle loss durante il training (valori medi per epoca).

Epoch	D_A	G_A	$\mathcal{L}_{cyc,A}$	D_B	G_B	$\mathcal{L}_{cyc,B}$
5	0.246	0.391	2.015	0.289	0.422	1.585
25	0.188	0.433	1.266	0.173	0.412	1.088
50	0.122	0.619	1.230	0.213	0.641	0.927
75	0.103	0.801	0.737	0.196	0.442	0.942
100	0.208	0.509	0.691	0.150	0.524	0.754
125	0.092	0.692	0.744	0.166	0.331	0.600
150	0.089	0.708	0.716	0.134	0.327	0.597
175	0.108	0.705	0.689	0.170	0.391	0.509
200	0.052	0.837	0.565	0.205	0.453	0.337

La cycle consistency loss mostra un andamento monotono decrescente per entrambe le direzioni, passando da valori iniziali di 2.015 (A) e 1.585 (B) a epoch 5 fino a 0.565 e 0.337 rispettivamente a epoch 200. Questo pattern, pur indicando che la rete impara a ricostruire le immagini attraverso il ciclo completo $A \to B \to A$, scende troppo rapidamente rispetto alla qualità visiva effettiva delle traduzioni, suggerendo un possibile overfitting sulla metrica di ricostruzione piuttosto che sulla qualità semantica.

Le loss dei discriminatori presentano comportamenti asimmetrici. D_A diminuisce costantemente da 0.246 a epoch 5 fino a 0.052 a epoch 200, raggiungendo valori criticamente bassi già a epoch 75 (0.103). Valori così bassi indicano che il discriminatore D_A classifica correttamente quasi tutte le immagini, sia reali che generate, suggerendo che G_B non riesce a produrre immagini sintetiche convincenti oppure che D_A è diventato troppo potente. Al contrario, D_B mantiene valori più stabili nell'intervallo $0.13 - 0.29$, sebbene mostri anch'esso una tendenza discendente.

Le loss dei generatori G_A e G_B presentano elevata variabilità durante tutto il training. G_A oscilla tra 0.391 a epoch 5 e 0.837 a epoch 200, con fluttuazioni significative (ad esempio da 0.619 a epoch 50 a 0.801 a epoch 75). Questa instabilità indica che il generatore fatica a trovare un equilibrio stabile con il discriminatore D_B. Similmente, G_B varia tra 0.327 e 0.641, con oscillazioni che persistono anche nelle fasi finali del training. Queste fluttuazioni sono sintomatiche di un training GAN instabile dove generatori e discriminatori non convergono verso un equilibrio.

Il problema più grave osservato è la perdita sistematica della struttura geometrica del tool durante la traduzione. Nelle immagini generate, i contorni netti e le superfici definite dei render Unity vengono progressivamente degradati, con bordi che diventano sfocati o completamente persi. In alcuni casi, parti significative del tool scompaiono o vengono fuse con lo sfondo, rendendo l'oggetto irriconoscibile. Questo fenomeno è

particolarmente evidente quando lo strumento si torva su uno sfondo con la stessa tonalità di colore.

Si osservano anche artefatti a scacchiera (*checkerboard artifacts*) che si manifestano come pattern regolari a griglia, particolarmente evidenti su sfondi monocromatici o uniformi. Gli artefatti a scacchiera sono causati dalla sovrapposizione disuniforme di kernel nella deconvoluzione. Soluzioni architetturali come l'uso di upsampling seguito da convoluzione standard invece di transposed convolution potrebbero mitigare questo problema, ma richiederebbero modifiche sostanziali all'implementazione.

Il mixing incontrollato di colori costituisce un'ulteriore criticità. Le immagini tradotte mostrano frequentemente mescolanze di tonalità che non corrispondono né al dominio sintetico né a quello reale. Le superfici degli strumenti, che nei render Unity presentano riflessi coerenti e texture uniformi, vengono trasformate in aree con gradazioni di colore innaturali.

La ricostruzione attraverso il ciclo completo $A \to B \to A$ mostra perdita di dettagli significativi rispetto all'immagine originale, nonostante i bassi valori di cycle consistency loss. Questo disallineamento tra metrica quantitativa e qualità percettiva evidenzia i limiti delle loss L_1 nel catturare proprietà semantiche delle immagini.

L'analisi congiunta dei dati quantitativi e qualitativi permette di identificare diverse cause delle problematiche osservate. Lo sbilanciamento tra discriminatori rappresenta un problema fondamentale: D_A diventa troppo forte troppo rapidamente (loss < 0.1 già a epoch 75), impedendo a G_B di migliorare. Questo sbilanciamento è probabilmente dovuto alla maggiore complessità del task di generare immagini sintetiche rispetto a immagini reali, dato che il dominio sintetico presenta caratteristiche più regolari e prevedibili.

La rapida discesa della cycle consistency loss non corrisponde a un effettivo miglioramento qualitativo, suggerendo che la rete ottimizza la metrica di ricostruzione pixelwise senza preservare proprietà semantiche come forma, bordi e texture coerenti. Questo fenomeno è noto in letteratura come 'shortcut learning' [16], dove la rete trova soluzioni che minimizzano la loss formale senza risolvere il problema sottostante.

Il dominio sintetico presenta caratteristiche difficili da mappare realisticamente: superfici perfettamente lisce, illuminazione uniforme, e assenza di imperfezioni. Il generator G_A deve imparare a introdurre variabilità realistica (graffi, usura, riflessi irregolari) che non è presente nel dominio sorgente. La variabilità limitata del dataset reale (512 immagini) potrebbe non fornire copertura sufficiente dello spazio delle possibili texture e condizioni di illuminazione.

Data la natura delle problematiche riscontrate, è utile considerare approcci alternativi per il domain adaptation. Il fine-tuning di modelli pre-addestrati su grandi dataset (ImageNet, COCO) con dati reali limitati ha mostrato risultati superiori in letteratura per task di detection e segmentation. Questo approccio sfrutta feature generali apprese su milioni di immagini naturali, richiedendo solo centinaia di esempi del dominio target per adattamento efficace.

L'augmentation avanzata del dataset sintetico rappresenta un'alternativa computazionalmente più efficiente. Tecniche come l'aggiunta di rumore realistico, simulazione

di motion blur, variazione di condizioni di illuminazione e inserimento di occlusioni parziali possono ridurre significativamente il domain gap senza necessità di GAN. Questi metodi sono deterministici, controllabili e non soffrono dei problemi di instabilità tipici del training GAN.

L'utilizzo diretto di dati sintetici con appropriate tecniche di domain randomization ha dimostrato efficacia in applicazioni robotiche. Variando sistematicamente parametri di rendering (texture, illuminazione, posizione camera), è possibile generare dataset sintetici sufficientemente diversificati da generalizzare al dominio reale, evitando completamente la necessità di traduzione esplicita.

L'esperimento con CycleGAN ha rivelato limitazioni fondamentali nell'applicazione di questo approccio al problema specifico di domain adaptation per tool industriali. Nonostante 200 epoch di training e ottimizzazione delle loss quantitative, la qualità visiva delle traduzioni rimane inadeguata per utilizzo pratico. Le problematiche principali includono perdita di struttura geometrica, artefatti visibili, instabilità del training e disallineamento tra metriche quantitative e qualità percettiva.

Tabella 3.7: Problematiche identificate e possibili soluzioni (non implementate).

Problematica	Possibile Soluzione
Sbilanciamento discriminatori	Learning rate differenziati; Spectral normalization
Artefatti a scacchiera	Upsample + Conv invece di TransposeConv
Perdita struttura geometrica	Aggiunta di perceptual loss; Feature matching
Instabilità training	Progressive growing; Gradient penalty
Dataset limitato	Data augmentation; Semi-supervised learning

Capitolo 4

Risultati

Questo capitolo presenta una valutazione quantitativa delle performance dei modelli $KP - PVNet_R$ e $BB - PVNet_R$ su un dataset reale di validazione. L'analisi è strutturata per confrontare sistematicamente l'accuratezza di predizione tra i due approcci, l'effetto delle condizioni ambientali controllate e le differenze di performance tra i tre strumenti chirurgici analizzati.

4.1 Dataset di validazione e protocollo sperimentale

I modelli per la predizione di keypoints ($KP - PVNet_R$) e vertici della bounding box 3D ($BB - PVNet_R$), allenati sul dataset sintetico e successivamente raffinati su dati reali, sono stati testati su un dataset di validazione acquisito mediante il setup sperimentale descritto nella Sezione 3.4.1.

Per valutare l'accuratezza e la capacità di generalizzazione dei modelli, sono state definite variabili ambientali controllate. Nello specifico, sono stati studiati tre livelli di distanza dalla camera: lontano (L, 50 cm), medio (M, 40 cm) e vicino (V, 30 cm). Inoltre, sono stati considerati due tipi di sfondo: pulito (P), caratterizzato da una parete monocromatica, e caotico (C), contenente elementi di disturbo visivo. Per ciascuna delle sei condizioni risultanti (LP, LC, MP, MC, VP, VC), lo strumento è stato posizionato in una posa fissa e sono state acquisite 10 immagini, permettendo di analizzare la consistenza delle predizioni in frame con caratteristiche simili.

Il dataset di test risultante comprende 180 immagini totali, distribuite equamente tra i tre strumenti chirurgici (60 immagini per strumento), ciascuna predetta da entrambi i modelli. Per ogni predizione sono stati calcolati gli errori euclidei lungo i tre assi cartesiani (X, Y, Z) come distanze tra le coordinate predette e il ground truth.

Le performance sono state quantificate mediante l'errore euclideo medio espresso in millimetri per ciascuna coordinata spaziale. Per ogni immagine, l'errore è stato calcolato come media degli errori degli 8 keypoints (o vertici) predetti. Le Figure 4.1–4.3 presentano grafici a barre che illustrano la distribuzione degli errori per keypoint e condizione ambientale, mentre le Figure 4.4–4.6 visualizzano gli errori bidimensionali (X-Y) mediante ellissi di errore. Si noti che scale differenti sono state utilizzate

per ciascuna coordinata al fine di evidenziare pattern specifici, a discapito della comparabilità visiva diretta tra assi.

Per valutare la significatività statistica delle differenze osservate, sono stati condotti test t di Student paired-samples tra $KP - PVNet_R$ e $BB - PVNet_R$ (N=180 coppie di predizioni sulla stessa immagine) e test t independent-samples per il confronto tra condizioni di sfondo (N=90 per gruppo). Il livello di significatività è stato fissato a $\alpha = 0.05$.

4.2 Risultati per strumento

4.2.1 Trapano chirurgico

La Tab. 4.1 riassume gli errori medi per il trapano chirurgico. $KP-PVNet_R$ presenta errori medi di 14.07 mm (X), 13.31 mm (Y) e 94.17 mm (Z), mentre $BB - PVNet_R$ registra 21.31 mm (X), 26.24 mm (Y) e 172.42 mm (Z). L'errore sulla coordinata Z risulta significativamente superiore rispetto alle coordinate planari per entrambi i modelli.

Tabella 4.1: Errori euclidei per il trapano chirurgico (media ± deviazione standard).

Modello	**X (mm)**	**Y (mm)**	**Z (mm)**
$KP - PVNet_R$	14.07 ± 17.41	13.31 ± 19.40	94.17 ± 109.16
$BB - PVNet_R$	21.31 ± 20.73	26.24 ± 24.02	172.42 ± 198.14
Δ (BB - KP)	+7.24	+12.93	+78.25
Incremento (%)	+51.5%	+97.1%	+83.1%

Note: Δ = differenza assoluta.

Il test t paired-samples conferma differenze statisticamente significative tra i due modelli per le coordinate X e Z, mentre la coordinata Y non presenta differenze significative (Tab. 4.2).

Tabella 4.2: Confronto statistico $KP - PVNet_R$ vs $BB - PVNet_R$ per il trapano (paired t-test, N=60).

Coordinata	**p-value**	**t-statistic**	**Significatività**
X	0.0008	-3.5380	***
Y	0.5413	0.6145	ns
Z	0.0043	-2.9722	**

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo ($p \geq 0.05$).

L'analisi per condizione (Fig. 4.1) rivela errori minimi nelle configurazioni L e M (entrambi gli sfondi), con valori che si mantengono sotto i 30 mm per le coordinate planari. Le condizioni VP e VC presentano un incremento marcato, con errori sulla coordinata Z che raggiungono valori massimi superiori a 300 mm. La Fig. 4.4 evidenzia una maggiore dispersione lungo l'asse Y rispetto all'asse X, con ellissi di errore prevalentemente allungate verticalmente.

L'effetto dello sfondo presenta pattern differenziati tra i due modelli (Tab. 4.3). KP-PVNet mostra differenze statisticamente significative su tutte e tre le coordinate ($p<0.05$), con la coordinata Z che presenta la significatività più elevata ($p=0.003$, $t=-3.062$). BB-PVNet non evidenzia differenze significative per le coordinate planari X e Y ($p>0.05$), mentre la coordinata Z presenta una differenza altamente significativa ($p<0.001$, $t=-4.271$). I t-statistic negativi indicano che gli errori in condizioni di sfondo caotico sono sistematicamente maggiori rispetto alle condizioni di sfondo pulito per entrambi i modelli.

Tabella 4.3: Effetto dello sfondo per il trapano chirurgico: confronto Pulito vs Caotico.

Modello	**Coord.**	**p-value**	**t-statistic**	**Significatività**
$KP-PVNet_R$	X	0.011	-2.639	**
	Y	0.013	-2.567	*
	Z	0.003	-3.062	**
$BB-PVNet_R$	X	0.057	-1.939	ns
	Y	0.133	-1.523	ns
	Z	9.5×10^{-5}	-4.271	***

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo.
t-statistic negativi indicano errori maggiori in condizioni di sfondo caotico.

Tutti gli 8 keypoints presentano pattern di errore consistenti tra loro, con incrementi proporzionali all'avvicinarsi dell'oggetto. La differenza tra $KP-PVNet_R$ e $BB-PVNet_R$ è relativamente costante attraverso i diversi keypoints, con incrementi medi di circa 10 mm per X e Y, e 60 mm per Z.

4.2.2 Strumento di taglio

La Tab. 4.4 presenta gli errori medi per lo strumento di taglio. KP-PVNet registra errori di 20.21 mm (X), 13.59 mm (Y) e 22.71 mm (Z), con valori comparabili tra le tre coordinate. BB-PVNet mostra incrementi sostanziali: 30.66 mm (X), 23.24 mm (Y) e 68.39 mm (Z).

Tabella 4.4: Errori euclidei per lo strumento di taglio (media ± deviazione standard).

Modello	**X (mm)**	**Y (mm)**	**Z (mm)**
$KP-PVNet_R$	20.21 ± 18.23	13.59 ± 8.81	22.71 ± 14.17
$BB-PVNet_R$	30.66 ± 22.10	23.24 ± 16.73	68.39 ± 59.93
Δ (BB - KP)	+10.45	+9.65	+45.68
Incremento (%)	+51.7%	+71.0%	+201.1%

Note: Δ = differenza assoluta.

L'analisi di $KP-PVNet_R$ (Fig. 4.2) non evidenzia differenze marcate tra le tre coordinate nelle condizioni L e M. La condizione VP presenta un outlier significativo per la coordinata X (>100 mm), attribuibile al keypoint N.3 (base dello strumento), mentre gli altri keypoints mantengono errori inferiori a 35 mm nella stessa condizione.

$BB - PVNet_R$ mostra maggiore variabilità, con picchi di errore più frequenti nelle condizioni di sfondo caotico. Si osserva un trend crescente nelle configurazioni VP e VC per tutte le coordinate.

Il test t paired-samples mostra differenze per le coordinate X e Z, mentre la coordinata Y non presenta differenze significative, come per il trapano chirurgico. (Tab. 4.5).

Tabella 4.5: Confronto statistico $KP-PVNet_R$ vs $BB-PVNet_R$ per lo strumento di taglio (paired t-test, N=60).

Coordinata	p-value	t-statistic	Significatività
X	0.0018	-3.2731	***
Y	0.1796	-1.3581	ns
Z	0.0011	-3.4335	**

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo ($p \geq 0.05$).

$KP - PVNet_R$ non mostra alcuna differenza statisticamente significativa tra condizioni di sfondo pulito e caotico per nessuna delle tre coordinate (p>0.05 per tutte), con valori di p-value compresi tra 0.593 e 0.805 e t-statistic vicini allo zero. $BB - PVNet_R$, al contrario, evidenzia differenze altamente significative su tutte le coordinate (p<0.01), con p-value compresi tra 0.001 (X) e 0.005 (Z). I t-statistic sono consistentemente negativi e di magnitudine elevata (range: -2.956 a -3.394), indicando incrementi sistematici dell'errore in presenza di sfondo caotico.

Tabella 4.6: Effetto dello sfondo per lo strumento di taglio: confronto Pulito vs Caotico.

Modello	Coord.	p-value	t-statistic	Significatività
$KP - PVNet_R$	X	0.805	0.248	ns
	Y	0.719	0.362	ns
	Z	0.593	0.536	ns
$BB - PVNet_R$	X	0.001	-3.394	**
	Y	0.002	-3.299	**
	Z	0.005	-2.956	**

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo.

Lo strumento di taglio presenta la distribuzione di errori più omogenea tra i tre strumenti analizzati, con minore variabilità intra-keypoint. Le visualizzazioni bidimensionali (Fig. 4.5) mostrano ellissi di errore relativamente contenute per le condizioni L e M, con incremento sostanziale solo nelle configurazioni V.

4.2.3 Strumento di puntamento

Lo strumento di puntamento presenta gli errori più elevati tra i tre dispositivi analizzati (Tab. 4.7). $KP - PVNet_R$ registra 38.92 mm (X), 31.54 mm (Y) e 197.89 mm (Z), mentre $BB - PVNet_R$ raggiunge 57.72 mm (X), 56.86 mm (Y) e 282.54 mm (Z).

Tabella 4.7: Errori euclidei per lo strumento di puntamento (media ± deviazione standard).

Modello	**X (mm)**	**Y (mm)**	**Z (mm)**
$KP - PVNet_R$	38.92 ± 26.86	31.54 ± 25.34	197.89 ± 121.74
$BB - PVNet_R$	57.72 ± 61.44	56.86 ± 52.31	282.54 ± 192.90
Δ (BB - KP)	+18.80	+25.32	+84.65
Incremento (%)	+48.3%	+80.3%	+42.8%

Note: Δ = differenza assoluta.

L'analisi di $KP - PVNet_R$ per la coordinata X (Fig. 4.3) mostra variabilità elevata nelle condizioni LP, LC e VC. Per $BB - PVNet_R$, la configurazione VP con sfondo pulito presenta le performance peggiori. La coordinata Y conferma il trend di incremento dell'errore con la diminuzione della distanza, già osservato negli altri strumenti.

La coordinata Z raggiunge valori critici, con errori medi di 197.89 mm ($KP - PVNet_R$) e 282.54 mm ($BB - PVNet_R$). Singole predizioni nelle condizioni MP e VP superano i 600 mm di errore.

Tabella 4.8: Confronto statistico $KP-PVNet_R$ vs $BB-PVNet_R$ per lo strumento di puntamento (paired t-test, N=60).

Coordinata	**p-value**	**t-statistic**	**Significatività**
X	0.0001	-4.0658	***
Y	0.0022	-3.2051	**
Z	0.0467	-2.0318	*

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo ($p \geq 0.05$).

$KP - PVNet_R$ presenta un p-value borderline sulla coordinata X (p=0.054, t=-1.963), appena al di sopra della soglia di significatività convenzionale, mentre le coordinate Y e Z mostrano p-value elevati (0.817 e 0.989 rispettivamente). $BB - PVNet_R$ evidenzia p-value uniformemente compresi tra 0.083 e 0.125 per tutte le coordinate, con t-statistic positivi (range: 1.558 a 1.765). Sebbene nessun confronto raggiunga significatività statistica, i t-statistic di $BB - PVNet_R$ suggeriscono una tendenza consistente, mentre quelli di $KP - PVNet_R$ sono eterogenei e prossimi allo zero per Y e Z.

Le visualizzazioni bidimensionali (Fig. 4.6) evidenziano le ellissi di errore più estese tra i tre strumenti, con aree che riflettono la maggiore incertezza nella localizzazione. La geometria complessa del puntatore, caratterizzata da marker sferici, contribuisce all'incremento dell'errore rispetto agli strumenti con geometria più semplice.

4.2.4 Pattern generali osservati

L'analisi evidenzia tre pattern consistenti:

Tabella 4.9: Effetto dello sfondo per lo strumento di puntamento: confronto Pulito vs Caotico.

Modello	**Coord.**	**p-value**	**t-statistic**	**Significatività**
$KP-PVNet_R$	X	0.054	-1.963	ns
	Y	0.817	0.232	ns
	Z	0.989	-0.014	ns
$BB-PVNet_R$	X	0.125	1.558	ns
	Y	0.108	1.632	ns
	Z	0.083	1.765	ns

Note: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$; ns = non significativo.

1. **Effetto della distanza**: Per tutti gli strumenti e modelli, l'errore aumenta sistematicamente con l'avvicinarsi dell'oggetto alla camera, con incrementi particolarmente marcati nelle condizioni V (30 cm).

2. **Superiorità di KP-PVNet**: La predizione diretta di keypoints anatomici produce errori inferiori rispetto alla predizione dei vertici della bounding box in tutti i casi esaminati.

3. **Anisotropia dell'errore**: La coordinata Z presenta errori sistematicamente maggiori rispetto alle coordinate planari (X, Y) per entrambi i modelli, con rapporti Z/(X+Y) compresi tra 2× e 10× a seconda dello strumento.

4.3 Analisi del domain gap (sim-to-real transfer)

Il domain gap rappresenta la discrepanza tra le performance del modello su dati sintetici (training) e dati reali (testing). Questa sezione analizza quantitativamente il trasferimento di conoscenza dal dominio sintetico generato in Unity al dominio reale acquisito con Intel RealSense D435i.

Per quantificare il sim-to-real gap, sono state confrontate le performance su due test set distinti: un test set sintetico, creato replicando digitalmente il setup sperimentale di laboratorio descritto nella Sezione 3.4.1, e il test set reale contenente le 180 immagini acquisite fisicamente. Il test set sintetico è stato generato posizionando i modelli CAD degli strumenti nelle stesse pose, distanze e condizioni di sfondo del setup reale, utilizzando i medesimi parametri intrinseci della camera calibrati sperimentalmente.

Sono stati valutati i modelli base addestrati esclusivamente su dati sintetici (indicati come KP-PVNet e BB-PVNet), e le versioni raffinate tramite fine-tuning su immagini reali (indicate come KP-PVNet$_R$ e BB-PVNet$_R$). Questa configurazione permette di quantificare sia il gap iniziale dovuto al trasferimento diretto dal sintetico, sia l'efficacia del fine-tuning nel ridurre tale discrepanza.

La Tab. 4.10 presenta i risultati comparativi per il modello di predizione dei keypoints. Il modello addestrato esclusivamente su dati sintetici raggiunge un F1-Score medio di 0.9175 ± 0.0932 e PCK@0.05 del 95.7%. Tuttavia, il trasferimento

al dominio reale porta il F1-Score a scnedere a 0.1820 ± 0.2384, con decremento assoluto di 0.7355 punti (degradazione dell'80.2%). Il PCK@0.05 si riduce dall'95.7% all'11%. Il fine-tuning fa salire il F1-Score di 0.6985 ± 0.1222 sul test set reale, con incremento di 0.5165 punti rispetto al modello base (+284% relativo). La deviazione standard si riduce a 0.1222, indicando maggiore stabilità predittiva. Il PCK@0.05 sale al 69%.

Tabella 4.10: Confronto performance sintetico vs reale per KP-PVNet.

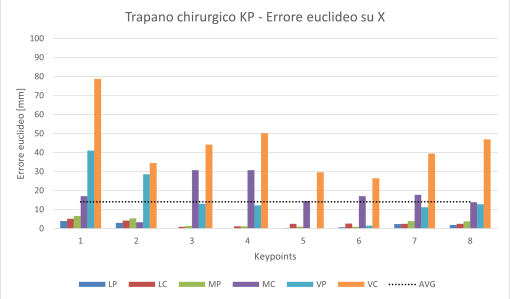
Metrica	**Modello**	**Test sintetico**	**Test reale**
F1-Score	KP-PVNet	0.9175 ± 0.0932	0.1820 ± 0.2384
	KP-PVNet$_R$	—	0.6985 ± 0.1222
PCK@0.05	KP-PVNet	95.7%	11%
	KP-PVNet$_R$	—	69%

La Tab. 4.11 riporta i risultati per il modello di predizione dei vertici della bounding box 3D. Sul test set sintetico ottiene F1-Score di 0.8840 ± 0.1792 e PCK@0.05 dell'88.1%, valori leggermente inferiori rispetto a KP-PVNet. Il domain gap per BB-PVNet risulta comparabile in magnitudine a quello di KP-PVNet. Sul test set reale, l'F1-Score del modello base precipita a 0.1533 ± 0.2047 (degradazione del 82.7%), mentre il PCK@0.05 scende al 9% (degradazione di 79.2 punti percentuali). Questi valori sono marginalmente inferiori a quelli di KP-PVNet (ΔF1 = -0.0287, ΔPCK = -2pp). Dopo fine-tuning, BB-PVNet$_R$ raggiunge F1-Score di 0.7196 ± 0.1448 e PCK@0.05 del 65% sul test set reale. Il recupero in termini assoluti è simile a quello di KP-PVNet (ΔF1 = +0.5663, ΔPCK = +56pp). Il gap residuo rispetto al sintetico rimane elevato (ΔF1 = 0.164, ΔPCK = 23.1pp).

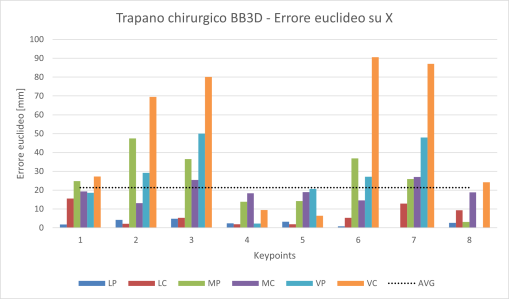
Questi valori indicano che il fine-tuning recupera circa il 70% del gap iniziale per entrambi gli approcci, lasciando un gap residuo del 25-30% rispetto alle performance sintetiche ideali.

Tabella 4.11: Confronto performance sintetico vs reale per BB-PVNet.

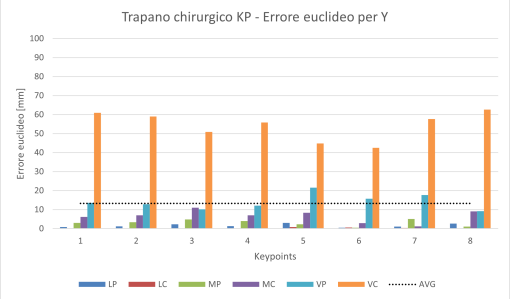
Metrica	**Modello**	**Test sintetico**	**Test reale**
F1-Score	BB-PVNet	0.8840 ± 0.1792	0.1533 ± 0.2047
	BB-PVNet$_R$	—	0.7196 ± 0.1448
PCK@0.05	BB-PVNet	88.1%	9%
	BB-PVNet$_R$	—	65%



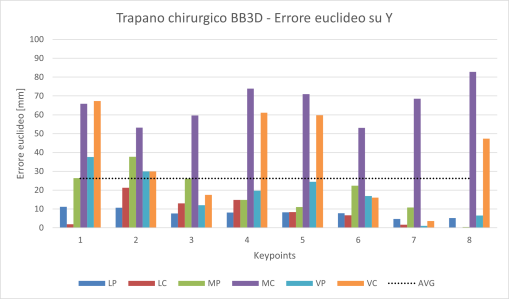
(a) Predizione di keypoints X.



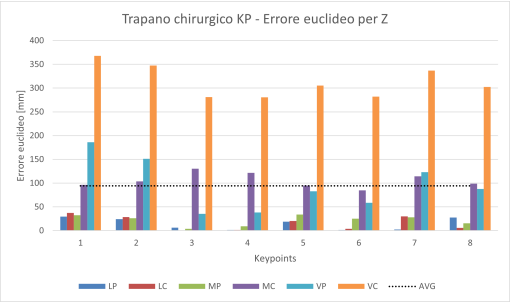
(b) Predizione bounding box 3D X.



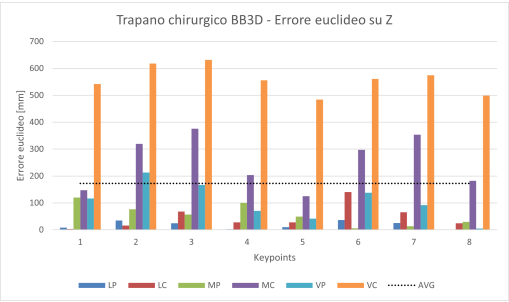
(c) Predizione di keypoints Y.



(d) Predizione bounding box 3D Y.

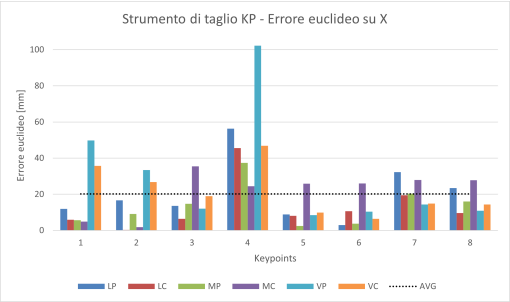


(e) Predizione di keypoints Z.

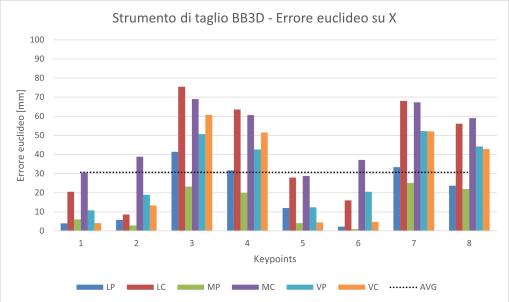


(f) Predizione bounding box 3D Z.

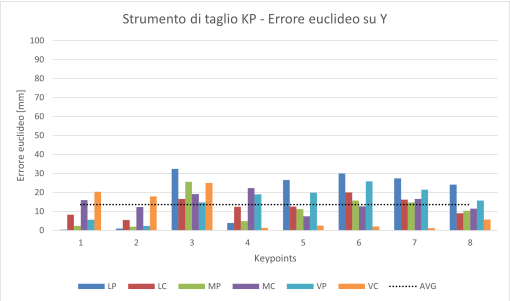
Figura 4.1: Visualizzazione dei risultati di predizione per il trapano chirurgico tramite grafici a barre. Sulle ordinate è riportato l'errore di posizionamento in mm, sulle ascisse gli otto keypoints o vertici predetti.



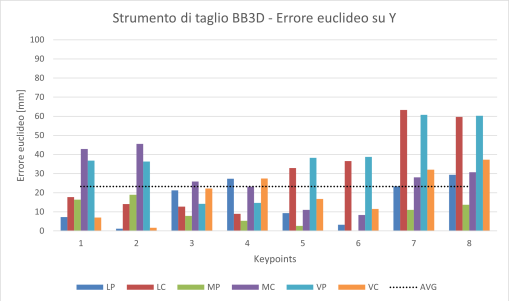
(a) Predizione di keypoints X.



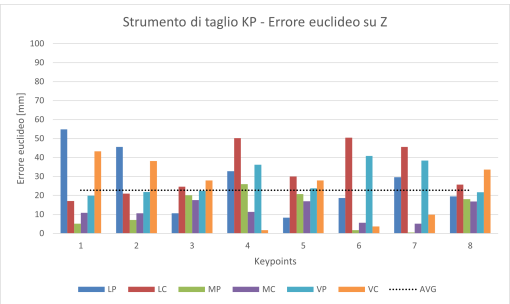
(b) Predizione bounding box 3D X.



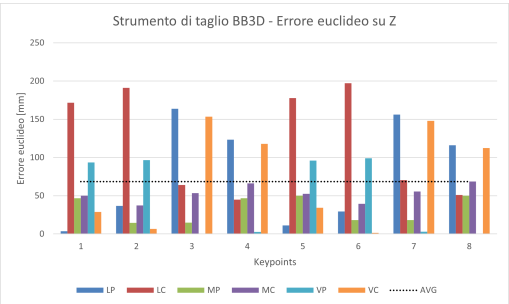
(c) Predizione di keypoints Y.



(d) Predizione bounding box 3D Y.

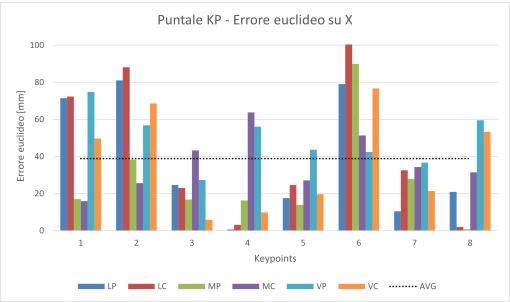


(e) Predizione di keypoints Z.

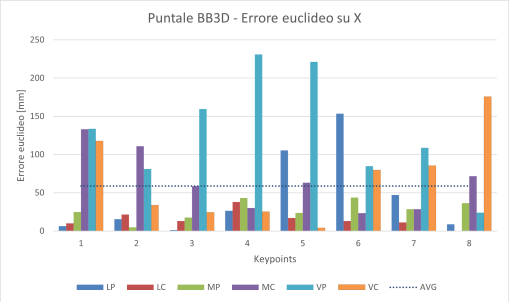


(f) Predizione bounding box 3D Z.

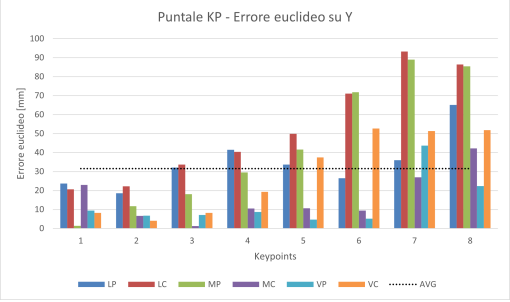
Figura 4.2: Visualizzazione dei risultati di predizione per lo strumento di taglio tramite grafici a barre. Sulle ordinate è riportato l'errore di posizionamento in mm, sulle ascisse gli otto keypoints o vertici predetti.



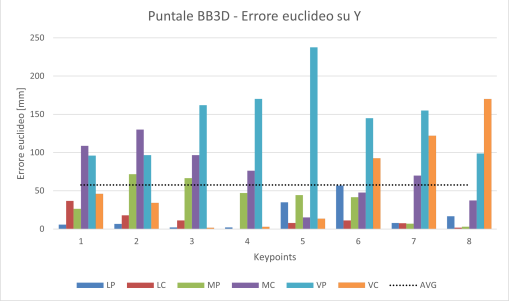
(a) Predizione di keypoints X.



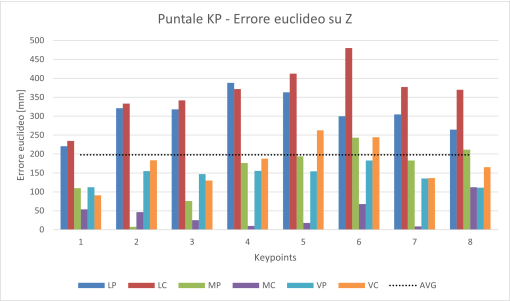
(b) Predizione bounding box 3D X.



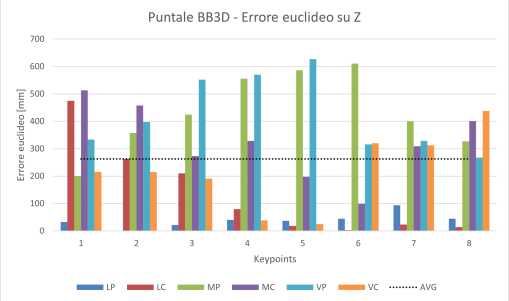
(c) Predizione di keypoints Y.



(d) Predizione bounding box 3D Y.

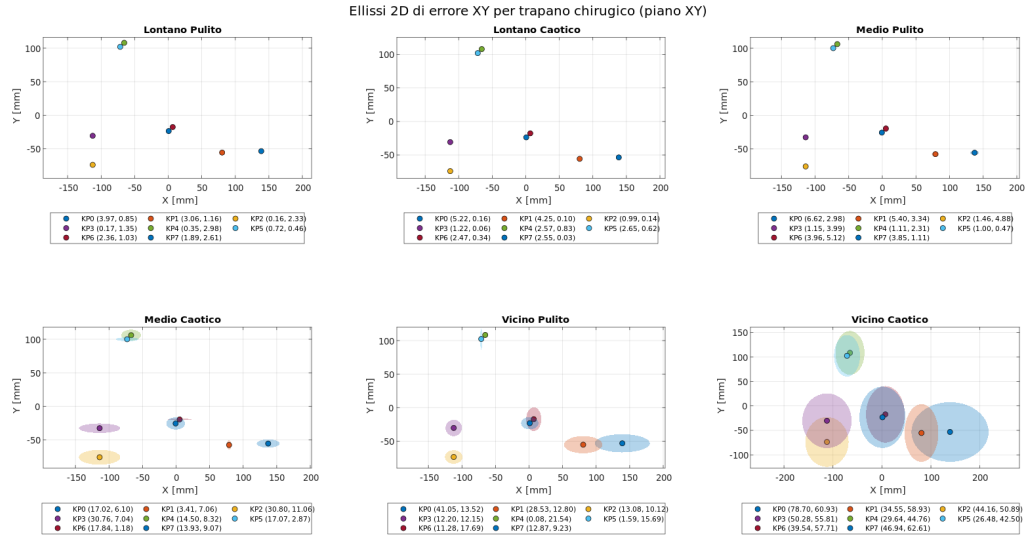


(e) Predizione di keypoints Z.

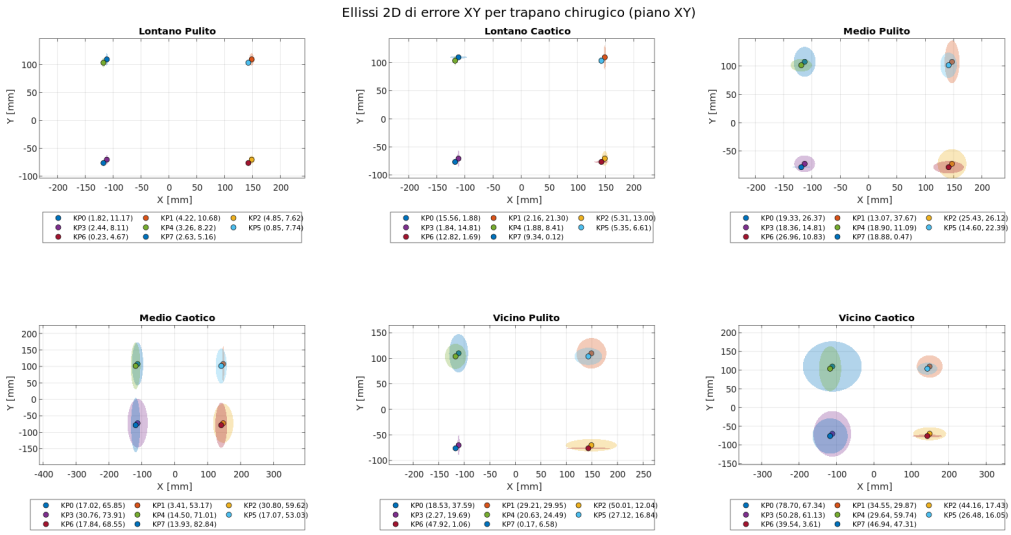


(f) Predizione bounding box 3D Z.

Figura 4.3: Visualizzazione dei risultati di predizione per lo strumento di puntamento tramite grafici a barre. Sulle ordinate è riportato l'errore di posizionamento in mm, sulle ascisse gli otto keypoints o vertici predetti.

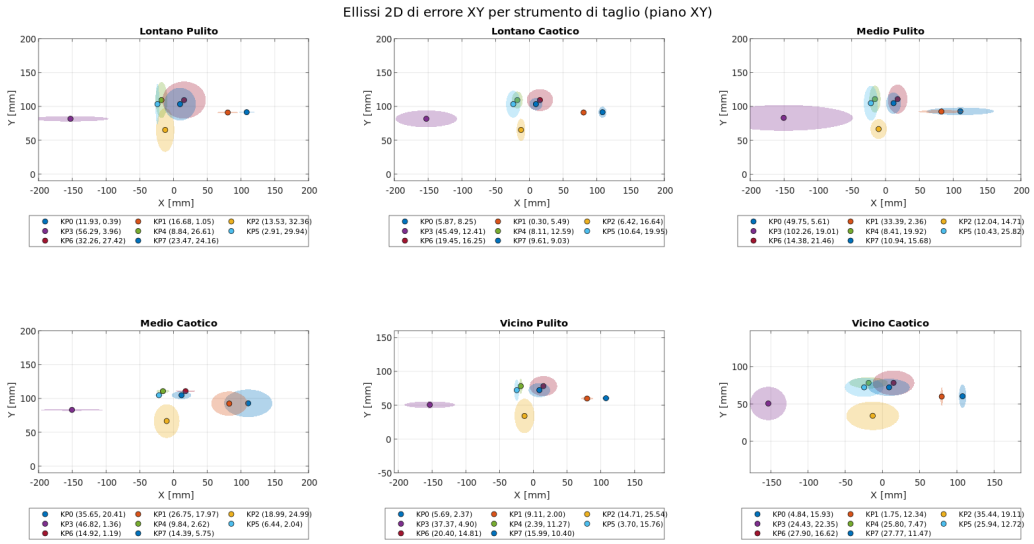


(a) Predizione di keypoints.

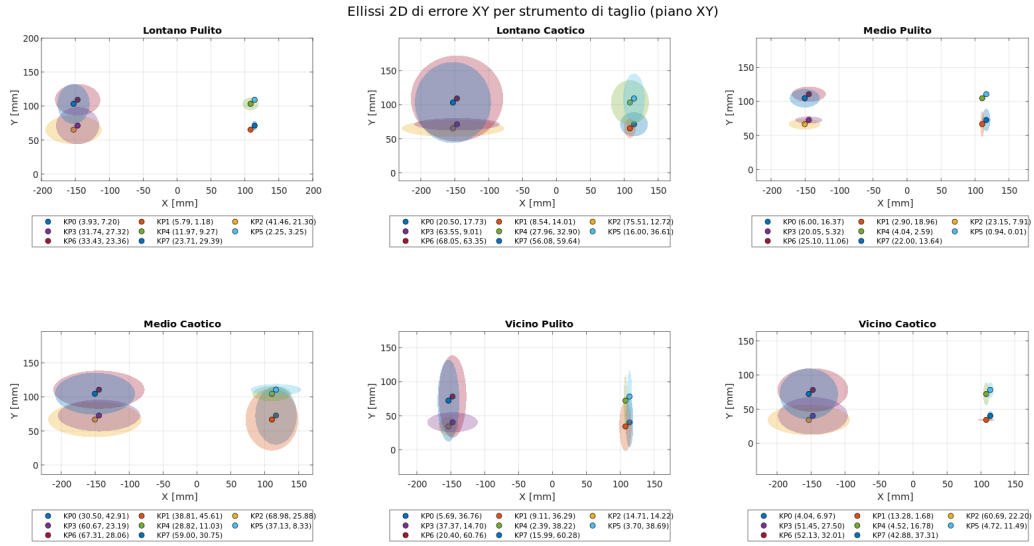


(b) Predizione della bounding box 3D.

Figura 4.4: Visualizzazione bidimensionale degli errori di predizione per il **trapano chirurgico**. Le ellissi rappresentano la distribuzione degli errori sul piano X-Y per ciascun keypoint nelle diverse condizioni ambientali. L'assenza di ellissi visibili indica errori inferiori alla dimensione del marker.

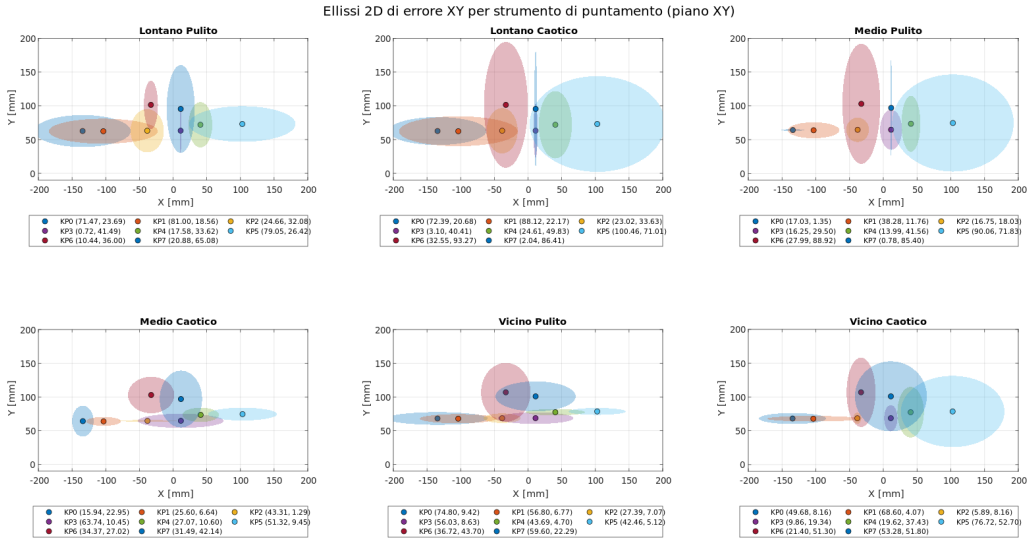


(a) Predizione di keypoints.

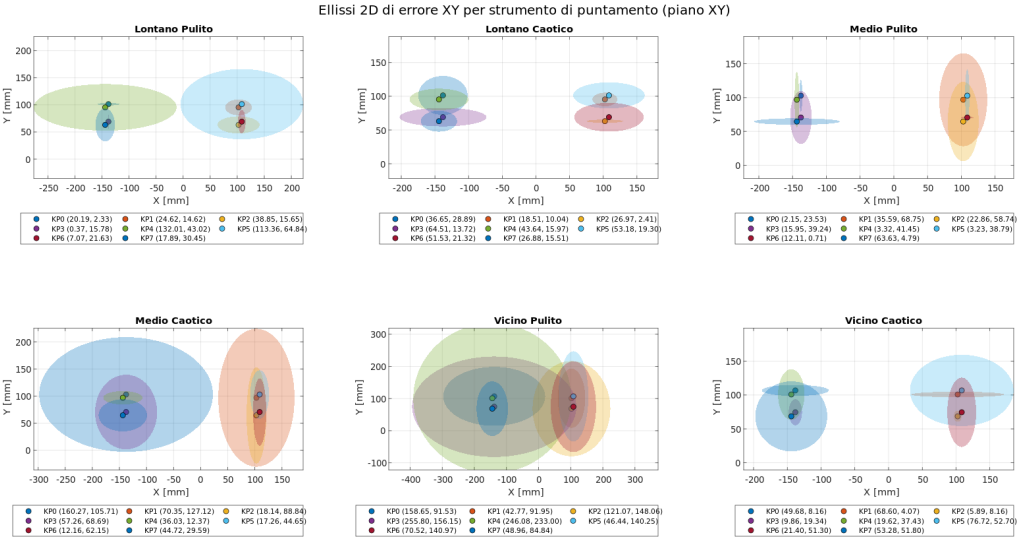


(b) Predizione della bounding box 3D.

Figura 4.5: Visualizzazione bidimensionale degli errori di predizione per lo **strumento di taglio**. Le ellissi rappresentano la distribuzione degli errori sul piano X-Y per ciascun keypoint nelle diverse condizioni ambientali.



(a) Predizione di keypoints.



(b) Predizione della bounding box 3D.

Figura 4.6: Visualizzazione bidimensionale degli errori di predizione per lo **strumento di puntamento**. Le ellissi rappresentano la distribuzione degli errori sul piano X-Y per ciascun keypoint nelle diverse condizioni ambientali. Le aree maggiori riflettono la maggiore incertezza nella localizzazione di questo strumento.

Capitolo 5

Discussione

Il presente lavoro si è posto l'obiettivo di sviluppare e validare una pipeline end-to-end per la generazione automatica di dati sintetici e la stima della posa 6DoF di strumenti chirurgici mediante tecniche di deep learning. L'approccio proposto si articola in due componenti principali: un sistema di generazione di dataset sintetici fotorealistici implementato in ambiente Unity Perception, capace di produrre automaticamente immagini RGB, maschere di segmentazione, mappe di profondità e annotazioni 2D/3D, e l'addestramento e validazione di due varianti del modello PVNet per la predizione rispettivamente di keypoints semantici (KP-PVNet) e vertici della bounding box 3D (BB-PVNet), con successiva stima della posa tramite algoritmo PnP.

L'analisi sperimentale condotta su 180 immagini reali acquisite in condizioni controllate ha permesso di quantificare le prestazioni effettive del sistema proposto e di identificare i principali fattori che influenzano l'accuratezza della stima di posa. I risultati ottenuti evidenziano una differenza di performance tra i due modelli sviluppati, con KP-PVNet che presenta errori euclidei medi sistematicamente inferiori rispetto a BB-PVNet su tutte e tre le coordinate spaziali per ciascuno dei tre strumenti analizzati. Per il trapano chirurgico, KP-PVNet registra errori medi di 14.07 mm (X), 13.31 mm (Y) e 94.17 mm (Z), mentre BB-PVNet mostra incrementi del 51.5%, 97.1% e 83.1% rispettivamente, raggiungendo 21.31 mm, 26.24 mm e 172.42 mm. Questa tendenza si conferma anche per lo strumento di taglio e per il puntatore marker-based, sebbene con magnitudini differenti.

La superiorità di KP-PVNet può essere attribuita al fatto che i keypoints semantici sono stati posizionati manualmente su feature geometriche distintive degli strumenti, come spigoli, estremità funzionali e punti di discontinuità morfologica, garantendo una distribuzione spaziale ottimizzata per la risoluzione del problema PnP. Al contrario, i vertici della bounding box 3D sono definiti matematicamente come gli estremi del parallelepipedo orientato che racchiude l'oggetto, senza alcuna considerazione delle caratteristiche visive o della distribuibilità locale. Durante il training, KP-PVNet può sfruttare feature locali distintive presenti in prossimità dei keypoints semantici, mentre BB-PVNet deve apprendere a localizzare punti che spesso si trovano nello spazio vuoto circostante l'oggetto, rendendo più difficile l'estrazione di feature rilevanti

dal contesto visivo.

Un pattern sistematico emerso dall'analisi è la marcata disparità tra l'accuratezza sulle coordinate planari (X, Y) e quella sulla coordinata di profondità (Z). Per tutti e tre gli strumenti e per entrambi i modelli, l'errore medio sulla coordinata Z risulta significativamente superiore, raggiungendo in alcuni casi valori fino a dieci volte maggiori rispetto alle coordinate nel piano immagine. Questo fenomeno è particolarmente evidente per il trapano chirurgico e il puntatore marker-based, dove gli errori medi su Z superano i 90 mm e i 190 mm rispettivamente per KP-PVNet, i quali sono anche gli strumenti che si sviluppano maggiormente in profondità, a differenza dello strumento di taglio che nella posizione nota varia di soli 2 cm sulla Z. Tale discrepanza è coerente con le proprietà intrinseche della proiezione prospettica e con le limitazioni note dei metodi basati su immagini monoculari RGB. [21]

Dal punto di vista geometrico, la relazione tra errore di localizzazione 2D nel piano immagine e errore di stima 3D lungo l'asse ottico non è lineare, ma dipende dalla distanza dell'oggetto dalla camera secondo un fattore quadratico. Un errore di pochi pixel nella localizzazione di un keypoint nell'immagine si traduce in un errore proporzionale sulla profondità secondo la relazione $\Delta Z \approx (Z^2/f) \cdot \Delta u$, dove f è la lunghezza focale e Δu l'errore in pixel [21]. Questo spiega perché, nelle condizioni sperimentali di distanza ravvicinata (30 cm per la configurazione V), anche piccole imprecisioni nella predizione dei vettori direzionali di PVNet generano errori sostanziali sulla coordinata Z. La sensibilità alla distanza emerge chiaramente dai risultati per condizione ambientale, dove le configurazioni V (vicino) presentano errori significativamente superiori rispetto alle configurazioni M (medio) e L (lontano) per entrambi i modelli.

Un ulteriore fattore che contribuisce all'incertezza sulla profondità è l'assenza di informazioni stereo o depth nel flusso di elaborazione. PVNet opera esclusivamente su immagini RGB monoculari, stimando la profondità attraverso feature monoculari come dimensione apparente dell'oggetto, prospettiva e occlusioni. Tuttavia, tali feature forniscono informazioni ambigue, specialmente per oggetti di dimensioni note osservati a distanze variabili.

L'analisi dell'effetto dello sfondo rivela comportamenti differenziati tra i due modelli e tra i diversi strumenti. Per il trapano chirurgico, KP-PVNet mostra differenze statisticamente significative su tutte e tre le coordinate quando confrontato tra condizioni di sfondo pulito e caotico, con p-value compresi tra 0.003 e 0.013 e t-statistic negativi che indicano incrementi sistematici dell'errore in presenza di elementi di disturbo visivo. BB-PVNet, invece, presenta significatività statistica solo sulla coordinata Z ($p < 0.001$), suggerendo una maggiore robustezza alle variazioni di sfondo per le coordinate planari ma maggiore sensibilità per la stima di profondità. Questo pattern si inverte completamente per lo strumento di taglio, dove KP-PVNet non mostra alcuna differenza significativa ($p > 0.05$ per tutte le coordinate, con p-value compresi tra 0.593 e 0.805), mentre BB-PVNet evidenzia differenze altamente significative su tutte e tre le coordinate ($p < 0.01$). Per il puntatore marker-based, nessuno dei due modelli raggiunge significatività statistica, sebbene BB-PVNet mostri

p-value borderline compresi tra 0.083 e 0.125.

Questa variabilità nell'effetto dello sfondo tra strumenti suggerisce che le caratteristiche geometriche e visive specifiche di ciascun oggetto modulano la sensibilità del modello agli elementi di disturbo. Lo strumento di taglio, caratterizzato da una geometria allungata e relativamente semplice con alta distintività visiva rispetto allo sfondo, permette a KP-PVNet di mantenere performance stabili indipendentemente dalla complessità della scena. Al contrario, il trapano chirurgico presenta una struttura più complessa degradando la qualità delle predizioni. BB-PVNet, predicendo punti nello spazio circostante piuttosto che sulla superficie dell'oggetto, risulta più vulnerabile a interferenze da oggetti adiacenti o pattern visivi complessi nello sfondo, come identificato già da [48].

Il puntatore marker-based rappresenta il caso più critico, presentando errori medi significativamente superiori rispetto agli altri due strumenti per entrambi i modelli. KP-PVNet registra 38.92 mm (X), 31.54 mm (Y) e 197.89 mm (Z), mentre BB-PVNet raggiunge 57.72 mm, 56.86 mm e 282.54 mm rispettivamente. La complessità geometrica del puntatore, caratterizzato da una struttura sottile a Y e quattro marker sferici disposti in configurazione tetraedrica, introduce sfide specifiche per l'apprendimento di rappresentazioni discriminative. Le superfici sferiche dei marker mancano di feature angolari o spigoli netti che faciliterebbero l'estrazione di keypoints robusti e la texture differente dal resto dello strumento ne complica la segmentazione.

La visualizzazione bidimensionale degli errori tramite ellissi conferma quantitativamente queste osservazioni. Per il puntatore, le ellissi di errore presentano aree significativamente maggiori rispetto agli altri strumenti, con dispersione prevalente lungo entrambi gli assi X e Y. Questa distribuzione isotropa dell'errore suggerisce incertezza generalizzata nella localizzazione piuttosto che bias direzionali, indicando che il modello non riesce a identificare consistentemente feature affidabili per il processo di voting. Al contrario, le ellissi per il trapano e lo strumento di taglio mostrano pattern più strutturati, spesso allungate preferenzialmente lungo una direzione specifica, riflettendo l'anisotropia delle geometrie degli strumenti e la maggiore affidabilità nella stima lungo certi assi.

L'analisi del domain gap evidenza una degradazione delle performance quando i modelli addestrati esclusivamente su dati sintetici vengono testati direttamente su immagini reali. KP-PVNet passa da un F1-Score di 0.9175 sul test set sintetico a 0.1820 sul test set reale, corrispondente a una degradazione dell'80.2%. Similmente, il PCK@0.05 crolla dal 95.7% all'11%, indicando che solo l'11% dei keypoints predetti si trova entro il 5% della diagonale della bounding box dalla posizione ground truth. BB-PVNet mostra un pattern comparabile, con F1-Score che scende da 0.8840 a 0.1533 e PCK@0.05 da 88.1% a 9%. Questo gap sostanziale conferma l'ipotesi che, nonostante gli sforzi di domain randomization implementati nella pipeline Unity Perception, permangono differenze distribuzionali significative tra dati sintetici e reali che il modello non riesce a generalizzare autonomamente. Le sorgenti principali di tale discrepanza possono essere identificate in diversi fattori fotometrici e geometrici. Le proprietà dei materiali simulati in Unity, sebbene basate su modelli

di rendering fisicamente plausibili (HDRP con ray tracing hardware-accelerated), non catturano completamente la complessità delle interazioni luce-materia osservate negli strumenti reali stampati in PLA. In particolare, le micro-texture superficiali generate dal processo di stampa 3D, i pattern di stratificazione visibili e le imperfezioni locali introducono variabilità ad alta frequenza spaziale che non è replicata nei modelli CAD perfettamente lisci utilizzati per il rendering. Analogamente, il modello di illuminazione sintetico, basato su una Directional Light con randomizzazione di intensità e temperatura colore, non cattura la complessità dell'illuminazione ambientale reale in laboratorio. Quest'ultima include componenti di luce diffusa da riflessioni multiple sulle pareti, contributi da sorgenti fluorescenti con spettri di emissione caratteristici e variazioni temporali dovute a fluttuazioni nella rete elettrica. Tali componenti generano pattern che differiscono da quelli simulati, costituendo features che il modello apprende durante il training ma che non generalizzano al dominio target. Il rumore del sensore rappresenta un ulteriore fattore non modellato nella pipeline sintetica. Le immagini generate da Unity sono noiseless, mentre le acquisizioni della Intel RealSense D435i presentano rumore fotometrico dipendente dall'ISO, rumore di quantizzazione e artefatti di compressione JPEG. Questi elementi introducono componenti stocastiche ad alta frequenza che alterano localmente i gradienti di intensità utilizzati dalla rete convoluzionale per l'estrazione di feature, potenzialmente degradando la qualità delle rappresentazioni apprese.

Il fine-tuning su 300 immagini reali annotate riduce significativamente il gap, portando l'F1-Score di KP-PVNet a 0.6985 e il PCK@0.05 al 69% sul test set reale. Questo corrisponde a un recupero di 0.5165 punti di F1-Score rispetto al modello base, equivalente a circa il 70% del gap iniziale. BB-PVNet mostra miglioramenti analoghi, con F1-Score che sale a 0.7196 e PCK@0.05 al 65%. Questi risultati dimostrano che un numero relativamente limitato di esempi reali è sufficiente per adattare le rappresentazioni apprese su larga scala nel dominio sintetico alle peculiarità distribuzionali del dominio target, confermando l'efficacia della strategia di transfer learning adottata. Tuttavia, permane un gap residuo significativo rispetto alle performance sintetiche ideali. KP-PVNet raggiunge 0.6985 contro 0.9175 atteso (delta di 0.219), mentre BB-PVNet raggiunge 0.7196 contro 0.8840 (delta di 0.164). Questo indica che il fine-tuning, sebbene efficace, non elimina completamente le discrepanze distribuzionali e che ulteriori strategie di domain adaptation potrebbero essere necessarie per colmare il divario rimanente. La minore deviazione standard osservata dopo fine-tuning (0.1222 per KP-PVNet contro 0.2384 del modello base) suggerisce che l'adattamento migliora principalmente la consistenza delle predizioni piuttosto che eliminare errori sistematici.

Confrontando i risultati ottenuti con lo stato dell'arte della pose estimation, emergono considerazioni rilevanti. PVNet, nel paper originale di Peng et al. [48], riporta performance di ADD pari a 86.3% sul dataset LINEMOD, mentre He et al. [23] con PVN3D raggiungono ADD pari a 91.8% su LINEMOD e 52.6% su YCB-Video [74]. Tuttavia, questi benchmark operano su oggetti con texture distintive e geometrie relativamente semplici, in condizioni di illuminazione controllata e con dataset di training

composti da decine di migliaia di immagini reali annotate. Il presente lavoro affronta uno scenario più sfidante, caratterizzato da strumenti chirurgici, addestramento su dataset completamente sintetico e validazione in condizioni ambientali variabili.

La scelta di utilizzare un dataset sintetico di 24000 immagini risulta limitata rispetto ai dataset real-world tipicamente impiegati per instance-level pose estimation. YCB-Video, ad esempio, contiene 133827 frame annotati, mentre T-LESS [28] include circa 48000 immagini acquisite con setup multi-camera. Questa disparità quantitativa spiega parzialmente le performance inferiori osservate rispetto ai metodi SOTA operanti in condizioni più favorevoli. Tuttavia, l'approccio proposto offre vantaggi significativi in termini di scalabilità e costo di annotazione, consentendo la generazione di dataset arbitrariamente grandi con annotazioni perfette in tempi ridotti.

Un aspetto rilevante emerso dall'analisi delle fasi di ablazione riguarda l'impatto della composizione del dataset sulle performance finali. I risultati mostrano che dataset privi di frame multi-strumento e distrattori geometrici producono modelli con metriche superiori rispetto a configurazioni più complesse. Questo contrasta parzialmente con l'intuizione comune secondo cui l'esposizione a scenari più variabili migliorerebbe la capacità di generalizzazione. L'apparente contraddizione può essere spiegata considerando che, nella fase di training iniziale, la rete deve prima apprendere le rappresentazioni fondamentali dell'oggetto target prima di poter gestire efficacemente complessità aggiuntive come occlusioni o presenza di distrattori. La strategia ottimale potrebbe quindi prevedere un curriculum learning dove il modello è inizialmente addestrato su scene semplici e progressivamente esposto a configurazioni più complesse.

La variazione nel numero di immagini del dataset (Fase 1) rivela un plateau di performance intorno a 1000-1500 immagini, oltre il quale i guadagni in F1-Score e PCK@0.05 diventano marginali. Questo suggerisce che, per il task specifico e le configurazioni di randomizzazione implementate, esiste un punto di saturazione oltre il quale l'aggiunta di campioni sintetici non introduce variabilità sufficiente per migliorare ulteriormente la generalizzazione. Tale osservazione ha implicazioni pratiche per l'ottimizzazione del trade-off tra costo computazionale di generazione e qualità del modello finale, indicando che investire risorse nella diversificazione qualitativa del dataset potrebbe essere più efficace rispetto all'incremento quantitativo puro.

Il sistema di validazione basato su supporto stampato in 3D con rotazioni discrete a passi di 45° non copre uniformemente lo spazio delle pose possibili, introducendo un bias verso configurazioni allineate agli assi principali. Sebbene questa scelta sia motivata da considerazioni pratiche di riproducibilità e accuratezza del ground truth, limita la capacità di valutare performance su orientamenti arbitrari che caratterizzerebbero uno scenario chirurgico reale. L'estensione futura del setup sperimentale dovrebbe includere un sistema di motion capture professionale (OptiTrack, Vicon) o un braccio robotico calibrato (KUKA, UR5) per acquisire ground truth sub-millimetrico su traiettorie continue. In secondo luogo, il numero di strumenti analizzati è limitato rispetto alla varietà di dispositivi presenti in una sala operatoria tipica. Sebbene

gli strumenti selezionati rappresentino categorie funzionali diverse (perforazione, taglio, navigazione marker-based) e presentino geometrie sufficientemente variate per testare la robustezza del metodo, l'estensione a un set più ampio di strumenti sarebbe necessaria per validare completamente la scalabilità dell'approccio. In particolare, strumenti con geometrie altamente articolate (pinze chirurgiche, forbici) o con componenti deformabili (cavi elettrochirurgici) introdurrebbero sfide aggiuntive che il framework corrente, basato su assunzione di rigidità dell'oggetto, non potrebbe gestire.

La scelta di operare esclusivamente su immagini RGB monoculari, sebbene coerente con applicazioni AR leggere basate su smartphone o HMD compatti, preclude l'utilizzo di informazioni geometriche esplicite che potrebbero migliorare significativamente l'accuratezza sulla coordinata Z. L'integrazione di informazioni depth da sensori RGB-D, analogamente a quanto proposto da DenseFusion e PVN3D, costituirebbe un'estensione naturale che sfrutterebbe appieno le capacità della Intel RealSense D435i utilizzata per le acquisizioni.

Un ulteriore limite riguarda l'assenza di valutazione in condizioni operative realistiche, con presenza di tessuti biologici, fluidi, strumenti multipli simultanei e occlusioni dinamiche generate dalle mani del chirurgo. Le condizioni controllate di laboratorio, con sfondi statici e illuminazione costante, costituiscono uno scenario idealizzato che sottovaluta le sfide di deployment in sala operatoria.

La strategia di fine-tuning adottata, sebbene efficace nel ridurre il domain gap, richiede comunque un certo numero di immagini reali annotate (300 nel presente studio). L'acquisizione e annotazione di questi dati, sebbene significativamente ridotta rispetto ad approcci puramente supervisionati, rappresenta ancora un costo non trascurabile in termini di manodopera specializzata. Tecniche di domain adaptation completamente unsupervised, basate ad esempio su self-supervised learning o su student-teacher frameworks, potrebbero potenzialmente eliminare questa dipendenza, rappresentando una direzione promettente per ricerche future.

Dal punto di vista dei dati sintetici, l'implementazione di tecniche di rendering avanzato come path tracing Monte Carlo o l'utilizzo di motori specializzati (NVIDIA Isaac Sim, Blender Cycles) potrebbe ridurre ulteriormente il domain gap fotometrico. L'integrazione di modelli fisicamente accurati per la simulazione di materiali complessi migliorerebbe il realismo visivo senza incrementare significativamente il costo computazionale. Inoltre, la generazione procedurale di imperfezioni superficiali aggiungerebbe variabilità ad alta frequenza spaziale che faciliterebbe il trasferimento al reale.

L'implementazione di un sistema AR completo richiederebbe l'integrazione della pipeline di pose estimation con componenti di rendering real-time e tracking temporale. Filtri di Kalman estesi o particle filters potrebbero essere impiegati per stabilizzare le predizioni frame-to-frame, sfruttando la coerenza temporale tipica di movimenti chirurgici controllati. L'integrazione con piattaforme AR commerciali (Microsoft HoloLens, Magic Leap) o custom permetterebbe di valutare la latenza end-to-end e l'usabilità in contesti operativi simulati.

Dal punto di vista metodologico, l'esplorazione di tecniche di active learning per la selezione ottimale di frame reali da annotare potrebbe ridurre ulteriormente il costo di fine-tuning. Diverse strategie sampling (uncertainty, diversity o adversarial) identificherebbero automaticamente i campioni più informativi, massimizzando il guadagno in performance per unità di sforzo di annotazione.

In conclusione, il presente lavoro dimostra la fattibilità di un approccio completamente basato su dati sintetici per la pose estimation di strumenti chirurgici, evidenziando sia i punti di forza che le limitazioni attuali. I risultati confermano che Unity Perception [2] costituisce un framework efficace per la generazione automatizzata di dataset fotorealistici con annotazioni perfette, riducendo drasticamente il costo e la complessità dell'acquisizione dati rispetto ad approcci tradizionali. Il modello KP-PVNet, addestrato su 24000 immagini sintetiche e raffinato su 300 immagini reali, raggiunge accuratezza sub-centimetrica sulle coordinate planari per due dei tre strumenti analizzati, sebbene la stima di profondità rimanga critica con errori nell'ordine di decine di millimetri.

Il domain gap quantificato tra performance sintetiche e reali sottolinea la necessità di strategie di domain adaptation, con il fine-tuning che emerge come soluzione pratica ed efficace per il recupero del 70% circa del gap iniziale. L'insuccesso dell'approccio CycleGAN evidenzia che tecniche generiche di style transfer non garantiscono preservazione di proprietà geometriche essenziali per task di pose estimation ad alta precisione, suggerendo che approcci domain-specific con vincoli geometrici espliciti potrebbero essere più appropriati.

Le direzioni future identificate, dall'estensione a scenari multi-oggetto all'integrazione di informazioni depth, dall'adozione di architetture transformer all'implementazione di sistemi AR completi, delineano un percorso di ricerca che potrebbe portare gradualmente a soluzioni clinicamente applicabili. La validazione in ambienti ospedalieri reali, sebbene al di fuori dello scopo del presente lavoro, costituirà il passaggio necessario per tradurre i risultati ottenuti in benefici tangibili per chirurghi e pazienti, contribuendo all'avanzamento delle tecnologie AR per la chirurgia guidata da immagini.

Bibliografia

[1] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby e Alex Mouzakitis. «A survey on 3d object detection methods for autonomous driving applications». In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pp. 3782–3795 (cit. a p. 2).

[2] Steve Borkman et al. «Unity perception: generate synthetic data for computer vision». In: *arXiv preprint arXiv:2107.04259* (2021) (cit. alle pp. 8, 71).

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton e Carsten Rother. «Learning 6d object pose estimation using 3d object coordinates». In: *European conference on computer vision*. Springer. 2014, pp. 536–551 (cit. a p. 9).

[4] Giacomo Castellarin et al. «High accuracy and learning curve improvement of augmented reality in total knee arthroplasty: A single-centre study on 157 patients». In: *Journal of Experimental Orthopaedics* 12 (2025). PMC12255951 (cit. a p. 2).

[5] Angel X Chang et al. «ShapeNet: An information-rich 3D model repository». In: *arXiv preprint arXiv:1512.03012* (2015) (cit. a p. 10).

[6] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl e Dieter Fox. «Self-supervised 6d object pose estimation for robot manipulation». In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3665–3671 (cit. a p. 2).

[7] Maximilian Denninger et al. «Blenderproc: Reducing the reality gap with photorealistic rendering». In: *16th Robotics: Science and Systems, RSS 2020, Workshops*. 2020 (cit. a p. 8).

[8] Yan Di, Ruida Zhang, Zhiqiang Lou, Fabian Manhardt, Xiangyang Ji, Nassir Navab e Federico Tombari. «GPV-Pose: Category-level Object Pose Estimation via Geometry-guided Point-wise Voting». In: *CVPR*. 2022, pp. 6781–6791 (cit. a p. 5).

[9] Alexey Dosovitskiy. «An image is worth 16x16 words: Transformers for image recognition at scale». In: *arXiv preprint arXiv:2010.11929* (2020) (cit. a p. 41).

[10] Adrian Elmi-Terander et al. «Augmented reality navigation with intraoperative 3D imaging vs fluoroscopy-assisted free-hand surgery for spine fixation surgery: a matched-control study comparing accuracy». In: *Scientific Reports* 10.1 (2020), p. 707. DOI: 10.1038/s41598-020-57693-5 (cit. a p. 2).

[11] Luis Fernandez, Viviana Avila e Luiz Gonçalves. *A generic approach for error estimation of depth data from (stereo and RGB-D) 3D sensors*. 2017 (cit. a p. 13).

[12] Martin A Fischler e Robert C Bolles. «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography». In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. alle pp. 15, 16).

[13] Parisis Gallos, Charalabos Georgiadis, Joseph Liaskos e John Mantas. «Augmented reality glasses and head-mounted display devices in healthcare». In: *Data, Informatics and Technology: An Inspiration for Improved Healthcare*. IOS Press, 2018, pp. 82–85 (cit. a p. 2).

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand e Victor Lempitsky. «Domain-adversarial training of neural networks». In: *Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030 (cit. a p. 7).

[15] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang e Hang-Fei Cheng. «Complete solution classification for the perspective-three-point problem». In: *IEEE transactions on pattern analysis and machine intelligence* 25.8 (2003), pp. 930–943 (cit. a p. 16).

[16] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge e Felix A Wichmann. «Shortcut learning in deep neural networks». In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673 (cit. a p. 50).

[17] Bradley Genovese, Steven Yin, Sohail Sareh, Michael Devirgilio, Laith Mukdad, Jessica Davis, Veronica J Santos e Peyman Benharash. «Surgical hand tracking in open surgery using a versatile motion sensing system: are we there yet?» In: *The American Surgeon* 82.10 (2016), pp. 872–875 (cit. a p. 2).

[18] Georgios Georgakis, Arsalan Mousavian, Alexander C Berg e Jana Kosecka. «Synthesizing training data for object detection in indoor scenes». In: *arXiv preprint arXiv:1702.07836* (2017) (cit. a p. 7).

[19] Afzal Godil, Roger Eastman, T Hong et al. «Ground truth systems for object recognition and tracking». In: *National Institute of Standards and Technology (NIST): Gaithersburg, MA, USA* (2013) (cit. a p. 6).

[20] Grand View Research. *Augmented Reality And Virtual Reality In Healthcare Market Report 2030*. Rapp. tecn. Report GVR-1-68038-855-8. Grand View Research, 2024. URL: https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-in-healthcare-market (cit. a p. 2).

[21] Richard Hartley e Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. alle pp. 12, 66).

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. a p. 30).

[23] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan e Jian Sun. «PVN3D: A Deep Point-wise 3D Keypoints Voting Network for 6DoF Pose Estimation». In: *CVPR*. 2020, pp. 11632–11641 (cit. alle pp. 4, 22, 68).

[24] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige e Nassir Navab. «Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes». In: *Asian conference on computer vision*. Springer. 2012, pp. 548–562 (cit. alle pp. 9, 16, 17).

[25] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart e Kurt Konolige. «On pre-trained image features and synthetic images for deep learning». In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018 (cit. a p. 7).

[26] Tomas Hodan, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother e Jiri Matas. «BOP challenge 2020 on 6D object localization». In: *European Conference on Computer Vision Workshops (ECCVW)*. 2020 (cit. a p. 18).

[27] Tomas Hodan et al. «Bop: Benchmark for 6d object pose estimation». In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 19–34 (cit. alle pp. 9, 22).

[28] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis e Xenophon Zabulis. «T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects». In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2017, pp. 880–888 (cit. alle pp. 10, 69).

[29] Sabera Hoque, Md Yasir Arafat, Shuxiang Xu, Ananda Maiti e Yuchen Wei. «A comprehensive review on 3D object detection and 6D pose estimation with deep learning». In: *IEEE Access* 9 (2021), pp. 143746–143770 (cit. alle pp. 2, 4).

[30] Takuya Ikeda, Suomi Tanishige, Ayako Amma, Michael Sudano, Hervé Audren e Koichi Nishiwaki. «Sim2real instance-level style transfer for 6d pose estimation». In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 3225–3232 (cit. a p. 7).

[31] Roman Kaskman, Sergey Zakharov, Ivan Shugurov e Slobodan Ilic. «HomebrewedDB: RGB-D dataset for 6D pose estimation of 3D objects». In: *IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2019, pp. 2767–2776 (cit. a p. 11).

[32] Alexander Kirillov et al. «Segment anything». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4015–4026 (cit. a p. 41).

[33] Vincent Lepetit, Francesc Moreno-Noguer e Pascal Fua. «EP n P: An accurate O (n) solution to the P n P problem». In: *International journal of computer vision* 81.2 (2009), pp. 155–166 (cit. alle pp. 16, 35).

[34] Zhigang Li, Gu Wang e Xiangyang Ji. «CDPN: Coordinates-Based Disentangled Pose Network for Real-Time RGB-Based 6-DoF Object Pose Estimation». In: *ICCV*. 2019, pp. 7677–7686 (cit. a p. 4).

[35] Jian Liu et al. «Deep learning-based object pose estimation: A comprehensive survey». In: *arXiv preprint arXiv:2405.07801* (2024) (cit. a p. 3).

[36] Jianhui Liu, Yukang Chen, Xiaoqing Ye e Xiaojuan Qi. «Ist-net: Prior-free category-level pose estimation with implicit space transformation». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 13978–13988 (cit. a p. 5).

[37] Mingsheng Long, Yue Cao, Jianmin Wang e Michael Jordan. «Learning transferable features with deep adaptation networks». In: *International Conference on Machine Learning (ICML)*. PMLR. 2015, pp. 97–105 (cit. a p. 7).

[38] David G Lowe. «Distinctive image features from scale-invariant keypoints». In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. a p. 3).

[39] Liang Ma et al. «Augmented reality guidance improves accuracy of orthopedic drilling procedures». In: *Scientific Reports* 14 (2024), p. 24305. DOI: 10.1038/s41598-024-76132-3 (cit. a p. 2).

[40] Pablo Martinez-Gonzalez, Sergiu Oprea, John Alejandro Castro-Vargas, Alberto Garcia-Garcia, Sergio Orts-Escolano, Jose Garcia-Rodriguez e Markus Vincze. «Unrealrox+: An improved tool for acquiring synthetic data from virtual 3d environments». In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8 (cit. a p. 8).

[41] Jean-Philippe Mercier, Chaitanya Mitash, Philippe Giguere e Abdeslam Boularias. «Learning object localization and 6d pose estimation from simulation and weakly labeled real images». In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3500–3506 (cit. a p. 7).

[42] Philippe Merloz, Jocelyne Troccaz, Hervé Vouaillat, Christian Vasile, Jérôme Tonetti, Ahmad Eid e Stéphane Plaweski. «Fluoroscopy-based navigation system in spine surgery». In: *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine* 221.7 (2007), pp. 813–820 (cit. a p. 2).

[43] Christian Moro, Zane Štromberga, Athanasios Raikos e Allan Stirling. «The effectiveness of virtual and augmented reality in health sciences and medical anatomy». In: *Anatomical sciences education* 10.6 (2017), pp. 549–559 (cit. a p. 2).

[44] Nathan Morrical, Jonathan Tremblay, Yunzhi Lin, Stephen Tyree, Stan Birchfield, Valerio Pascucci e Ingo Wald. «Nvisii: A scriptable tool for photorealistic image generation». In: *arXiv preprint arXiv:2105.13962* (2021) (cit. alle pp. 7, 10).

[45] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton e Dieter Fox. «6-dof grasping for target-driven object manipulation in clutter». In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6232–6238 (cit. a p. 2).

[46] Christos Papaioannidis, Vasileios Mygdalis e Ioannis Pitas. «Domain-translated 3D object pose estimation». In: *IEEE Transactions on Image Processing* 29 (2020), pp. 9279–9291 (cit. a p. 7).

[47] Nadia Payet e Sinisa Todorovic. «From contours to 3d object detection and pose estimation». In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 983–990 (cit. a p. 4).

[48] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou e Hujun Bao. «PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation». In: *CVPR*. 2019, pp. 4561–4570 (cit. alle pp. 3, 22, 29, 31, 33, 67, 68).

[49] Arnau Ramisa, Shrihari Vasudevan, David Aldavert, Ricardo Toledo e Ramon Lopez de Mantaras. «Evaluation of the sift object recognition method in mobile robots». In: *Artificial Intelligence Research and Development*. IOS Press, 2009, pp. 9–18 (cit. a p. 3).

[50] Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi. «You only look once: Unified, real-time object detection». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788 (cit. a p. 22).

[51] Pavel Rojtberg e Thomas Pöllabauer. «YCB-Ev 1.1: Event-vision dataset for 6DoF object pose estimation». In: *European Conference on Computer Vision*. Springer. 2024, pp. 1–13 (cit. a p. 6).

[52] Olaf Ronneberger, Philipp Fischer e Thomas Brox. «U-net: Convolutional networks for biomedical image segmentation». In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. a p. 31).

[53] Radu Bogdan Rusu, Nico Blodow e Michael Beetz. «Fast point feature histograms (FPFH) for 3D registration». In: *2009 IEEE international conference on robotics and automation*. IEEE. 2009, pp. 3212–3217 (cit. a p. 3).

[54] Ruben Sagues-Tanco, Luis Benages-Pardo, Gonzalo López-Nicolás e Sergio Llorente. «Fast synthetic dataset for kitchen object segmentation in deep learning». In: *IEEE Access* 8 (2020), pp. 220496–220506 (cit. a p. 7).

[55] Caner Sahin, Guillermo Garcia-Hernando, Juil Sock e Tae-Kyun Kim. «A review on object pose recovery: From 3D bounding box detectors to full 6D pose estimators». In: *Image and Vision Computing* 96 (2020), p. 103898 (cit. a p. 2).

[56] Hannah Schieber, Kubilay Can Demir, Constantin Kleinbeck, Seung Hee Yang e Daniel Roth. «Indoor synthetic data generation: A systematic review». In: *Computer Vision and Image Understanding* 240 (2024), p. 103907 (cit. a p. 7).

[57] Guanya Shi, Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, Fabio Ramos, Animashree Anandkumar e Yuke Zhu. «Fast uncertainty quantification for deep object pose estimation». In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021 (cit. a p. 10).

[58] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi e Andrew Fitzgibbon. «Scene coordinate regression forests for camera relocalization in RGB-D images». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2930–2937 (cit. a p. 18).

[59] Leonid Sigal, Alexandru O Balan e Michael J Black. «Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion». In: *International journal of computer vision* 87.1 (2010), pp. 4–27 (cit. a p. 5).

[60] Baochen Sun e Kate Saenko. «Return of frustratingly easy domain adaptation». In: *AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016 (cit. a p. 7).

[61] Bugra Tekin, Sudipta N Sinha e Pascal Fua. «Real-time seamless single shot 6d object pose prediction». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 292–301 (cit. a p. 22).

[62] Thang To, Jonathan Tremblay, Duncan McKay, Yukie Yamaguchi, Kirby Leung, Adrian Balanon, Jia Cheng, William Hodge e Stan Birchfield. «NDDS: NVIDIA deep learning dataset synthesizer». In: *Proceedings of the CVPR 2018 Workshop on Real World Challenges and New Benchmarks for Deep Learning in Robotic Vision, Salt Lake City, UT, USA*. 2018, pp. 18–22 (cit. a p. 8).

[63] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba e Pieter Abbeel. «Domain randomization for transferring deep neural networks from simulation to the real world». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30 (cit. a p. 6).

[64] Jonathan Tremblay, Thang To e Stan Birchfield. «Falling things: A synthetic dataset for 3D object detection and pose estimation». In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 2038–2041 (cit. a p. 10).

[65] Jonathan Tremblay et al. «Training deep networks with synthetic data: Bridging the reality gap by domain randomization». In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 969–977 (cit. a p. 6).

[66] Stephen Tyree, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Jeffrey Smith e Stan Birchfield. «6-DoF pose estimation of household objects for robotic manipulation: An accessible dataset and benchmark». In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 13081–13088 (cit. a p. 11).

[67] Eric Tzeng, Judy Hoffman, Kate Saenko e Trevor Darrell. «Adversarial discriminative domain adaptation». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7167–7176 (cit. a p. 7).

[68] Osamu Ukimura e Inderbir S Gill. «Image-fusion, augmented reality, and predictive surgical navigation». In: *Urologic Clinics* 36.2 (2009), pp. 115–123 (cit. a p. 2).

[69] Shinji Umeyama. «Least-squares estimation of transformation parameters between two point patterns». In: *IEEE Transactions on pattern analysis and machine intelligence* 13.4 (2002), pp. 376–380 (cit. a p. 5).

[70] Petr Vávra, Jan Roman, Pavel Zonča, Peter Ihnát, Martin Němec, Jayant Kumar, Nagy Habib e Ahmed El-Gendi. «Recent development of augmented reality in surgery: a review». In: *Journal of healthcare engineering* 2017.1 (2017), p. 4574172 (cit. a p. 2).

[71] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei e Silvio Savarese. «DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion». In: *CVPR*. 2019, pp. 3343–3352 (cit. alle pp. 4, 17, 22).

[72] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song e Leonidas J Guibas. «Normalized object coordinate space for category-level 6D object pose and size estimation». In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2642–2651 (cit. a p. 10).

[73] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song e Leonidas J Guibas. «Normalized object coordinate space for category-level 6d object pose and size estimation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2642–2651 (cit. alle pp. 5, 7).

[74] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan e Dieter Fox. «Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes». In: *arXiv preprint arXiv:1711.00199* (2017) (cit. alle pp. 10, 68).

[75] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva e Antonio Torralba. «Sun database: Large-scale scene recognition from abbey to zoo». In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 3485–3492 (cit. a p. 23).

[76] Congmin Yang, Zijian Zhao e Sanyuan Hu. «Image-based laparoscopic tool detection and tracking using convolutional neural networks: a review of the literature». In: *Computer Assisted Surgery* 25.1 (2020), pp. 15–28 (cit. a p. 2).

[77] Daniel Yang, Tarik Tosun, Benjamin Eisner, Volkan Isler e Daniel Lee. «Robotic grasping through combined image-based grasp proposal and 3d reconstruction». In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 6350–6356 (cit. a p. 2).

[78] Xuyuan Yang e Guang Jiang. «A practical 3D reconstruction method for weak texture scenes». In: *Remote Sensing* 13.16 (2021), p. 3103 (cit. a p. 3).

[79] Brayan S Zapata-Impata, Pablo Gil, Jorge Pomares e Fernando Torres. «Fast geometry-based computation of grasping points on three-dimensional point clouds». In: *International Journal of Advanced Robotic Systems* 16.1 (2019), p. 1729881419831846 (cit. a p. 2).

[80] Zhengyou Zhang. «A flexible new technique for camera calibration». In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2002), pp. 1330–1334 (cit. alle pp. 14, 43).

[81] Jun-Yan Zhu, Taesung Park, Phillip Isola e Alexei A Efros. «Unpaired image-to-image translation using cycle-consistent adversarial networks». In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2223–2232 (cit. a p. 7).

[82] Lu Zou, Zhangjin Huang, Naijie Gu e Guoping Wang. «6D-ViT: Category-level 6D object pose estimation via transformer-based instance representation learning». In: *IEEE Transactions on Image Processing* 31 (2022), pp. 6907–6921 (cit. a p. 5).

Dediche

Mi sento di dover fare un ringraziamento
a chi c'è stato e ora non c'è più
a chi ancora c'è
a chi resterà