Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea Magistrale in Informatica

Valutazione di reti neurali non supervisionate per la stima di profondità da singola immagine

Relatore: Prof. Domenico G. SORRENTI

Co-relatore: Dott. Matteo VAGHI

Relazione della prova finale di:
Carlo RADICE
Matricola 807159

Anno Accademico 2020-2021

Università degli Studi di Milano - Bicocca
Dipartimento di Informatica Sistemistica e Comunicazione
Informatics and Robotics for Automation Lab

Indice

Capitolo 1

Introduzione

Nel campo della robotica mobile, comprendere correttamente la conformazione dell'ambiente permette a sistemi autonomi di navigare in maniera sicura ed efficace. In particolare, avere a disposizione l'informazione geometrica 3D rende i robot in grado di percepire ostacoli o localizzarsi rispetto ad una mappa, permettendo di portare a compimento task come *Simultaneous Localization And Mapping* (SLAM), navigazione e riconoscimento di oggetti. Per acquisire questa conoscenza i robot mobili utilizzano diverse tipologie di sensori tra cui camere, sonar e laser scanner. La scelta di utilizzare un determinato tipo di sensore viene fatta tenendo conto del tipo di task che bisogna risolvere. Inoltre, altri parametri come la precisione della misurazione, il peso e il costo in fase di acquisto e di mantenimento influenzano la scelta di adottare un sensore rispetto ad un altro. I sonar sono caratterizzati da un campo di lavoro ridotto e producono dati molto rumorosi rendendo difficili compiti complessi come la ricostruzione tridimensionale dell'ambiente, per questo vengono utilizzati molto spesso come sensori di prossimità (ad esempio nelle automobili sono usati come sensori di parcheggio). Altri sensori sono le camere RGB-D che acquisiscono, oltre al dato RGB, una immagine di profondità della scena. La loro applicazione si limita ad ambienti indoor in quanto hanno un campo di misurazione limitato che non permette il loro utilizzo outdoor. I laser scanner, chiamati anche *Light Detection and Ranging* (LIDAR), permettono di determinare la distanza di un oggetto o di una superficie utilizzando un impulso luminoso. Ad oggi, pur nella loro non perfezione, rappresentano lo stato dell'arte per quanto riguarda la precisione nelle misurazioni e l'affidabilità in condizioni diverse. La robustezza del laser deriva dal fatto che lavora solo ad una frequenza ed emette un fascio luminoso che arriva concentrato su un target, generando un eco intenso anche a distanze elevate e su materiali con riflettività non tanto alta. Tuttavia, in ambito automotive a causa degli svantaggi legati al costo elevato e alla presenza di una componente meccanica

che li rende poco affidabili e durevoli nelle condizioni di lavoro di un autoveicolo, l'adozione su grande scala risulta molto difficoltosa. Nel settore automotive vengono utilizzati prevalentemente per scopi di ricerca ma sono presenti alcune iniziative sperimentali come l'azienda Waymo che, a Phoenix e a San Francisco, li utilizza in unione ad altri sensori su veicoli facenti parte di una flotta di robot taxi autonomi. Di conseguenza, grazie ai bassi costi, a un notevole campo di misurazione e ad un già presente utilizzo in una grande varietà di altre applicazioni, per ricavare l'informazione di profondità, vengono adottate camere monoculari o in configurazione stereo. Metodi tradizionali di stima di profondità come *Structure from Motion* (SfM) e visione stereoscopica sfruttano l'informazione data da diversi punti di vista utilizzando la tecnica di triangolazione per ricostruire la rappresentazione tridimensionale dell'ambiente. Anche se la ricostruzione ha una qualità inferiore rispetto alla relativa scansione LIDAR, è comunque abbastanza precisa per essere utilizzata in diverse applicazioni, come ad esempio la stima di spostamento di un robot nel tempo, chiamata *Visual Odometry* (VO). D'altra parte stimare la profondità da singola immagine è un problema ancora aperto in letteratura e costantemente soggetto a ricerca. Il rapido sviluppo negli ultimi anni del deep learning ha portato anche allo studio di *Deep Neural Networks* (DNN) in grado di effettuare task di image processing ottenendo risultati migliori dei metodi tradizionali presenti in letteratura. La ricerca in questo settore ha portato allo studio di DNN in grado di stimare la profondità da singole immagini. Nel corso del tempo sono stati proposti diversi approcci che hanno permesso di migliorare l'accuratezza delle predizioni fino ad ottenere performance promettenti anche in contesti molto complicati come la ricostruzione di scene urbane dinamiche. Gli approcci supervisionati si basano sull'esistenza di dataset contenenti coppie immagine e dato di profondità per supervisionare in fase di training la stima di profondità. Tuttavia, poiché è difficile acquisire grandi quantità di esempi etichettati con il dato di profondità, in letteratura sono emersi approcci alternativi che si basano su apprendimento non supervisionato, con il vantaggio di poter utilizzare un insieme ristretto di esempi etichettati solo per la fase di valutazione.

Lo scopo del presente lavoro di tesi è lo studio e lo sviluppo di un nuovo metodo di valutazione delle prestazioni di reti neurali che stimano la profondità di una scena avendo come informazione solo una singola immagine. In particolare, l'obiettivo è quello di irrobustire il metodo di valutazione utilizzato in letteratura che si basa sul confronto tra la predizione effettuata dalla rete neurale ed una mappa ground truth ottenuta da una singola scansione LIDAR. In dataset molto usati in letteratura, come KITTI, la singola scansione è piuttosto sparsa e, inoltre, la posizione del sensore LIDAR sulla piattaforma non permette di scansionare parte della scena ottenendo la

maggior parte dei punti racchiusa in un breve intervallo di distanze dall'osservatore. Per molti punti che vengono ricostruiti dalla rete non sono disponibili punti nella ground truth. Di conseguenza, quando si effettua la valutazione di queste reti, i risultati che si ottengono sono poco veritieri in quanto sezioni considerevoli della scena non sono valutabili. Per superare questo limite, in questo lavoro viene presentata una nuova metodologia di valutazione delle prestazioni che si basa sull'allineamento di più scansioni LIDAR per ottenere una ground truth più ricca di informazione in un intervallo di profondità maggiore. In questo modo, nella successiva valutazione della predizione sono presenti più punti in tutta la scena con la possibilità di confrontare il valore 3D ricostruito con punti anche a distanza di decine di metri. Il lavoro oggetto della tesi è stato svolto in diverse fasi.

1. La fase iniziale del progetto è stata dedicata a una ricerca all'interno della letteratura di lavori per affrontare il problema della ricostruzione 3D dell'ambiente da immagini. Sono stati studiati diversi approcci che sfruttano tecniche basate su reti neurali per ricavare il dato di profondità. In particolare, la scelta è ricaduta su lavori in grado di essere applicati in ambito automotive che sfruttano DNN non supervisionate per la stima di profondità da singola immagine.

2. Successivamente, si è effettuata una ricerca di dataset in grado di essere utilizzati per valutare in qualche forma la stima di profondità ottenuta dalle reti. In particolare, si sono ricercati dataset che contenessero al loro interno sequenze di immagini, stereo o monculari e il dato LIDAR da utilizzare come ground truth per effettuare il confronto. Tra i vari dataset presenti in letteratura si è scelto di usare KITTI[13], in quanto considerato tra i principali benchmark per la stima di profondità e Oxford Robotcar [30]. Infatti, entrambi i dataset contengono al loro interno dati immagine e scansioni LIDAR. KITTI è un dataset molto utilizzato in letteratura per il task della stima di profondità sia in lavori supervisionati che non supervisionati. Il sensore velodyne a 64 piani di scansione permette di ottenere molti punti da riproiettare nell'immagine per effettuare la valutazione della predizione, tuttavia la sua posizione sulla piattaforma non gli permette di acquisire punti a intervalli di distanza elevati. KITTI-360 [26] utilizza gli stessi sensori camera e LIDAR di KITTI ma viene montato un GPS più preciso. Oxford Robotcar nella sua prima versione contiene al suo interno solo scansioni LIDAR a singolo e a 4 piani, non sufficienti per qualità ad essere utilizzate come ground truth. D'altra parte Oxford Robotcar Radar [1] contiene laser a 32 piani in grado di acquisire scansioni con un grado di precisione tale da poter essere utilizzate per la ground truth.

3. Si è proceduto a selezionare alcune tra le reti neurali deep viste in letteratura per essere dapprima allenate e poi valutate utilizzando l'approccio dello stato dell'arte sui dataset scelti. Per allenare le reti con KITTI viene utilizzato lo split dati definito da Eigen e Fergus [9] in quanto è alla base dello stato dell'arte per questo dataset. Poichè Oxford non viene utilizzato di norma per task di stima di profondità viene definito uno split di training che consiste nell'unione di immagini provenienti da 50 sequenze di Oxford Robotcar. Per valutare le performance delle reti vengono utilizzati sia lo split di test di Eigen e Fergus per KITTI sia dati immagine scelti tra le sequenze di KITTI-360. Viene usato Oxford Robotcar Radar come dataset di valutazione per reti allenate con Oxford Robotcar, in quanto contiene il dato LIDAR a 32 piani.

4. La fase successiva del progetto si è incentrata sulla creazione di una pipeline di valutazione basata sull'idea innovativa di concatenare scansioni LIDAR per ottenere una mappa di profondità ground truth densa, in modo tale da avere punti ad intervalli di distanza diversi. La pipeline si può riassumere in tre step: allineamento e registrazione delle scansioni; proiezione dei punti nel piano immagine; filtraggio delle occlusioni. La fase finale comprende la valutazione della predizione.

5. Per quanto concerne il primo step, vengono inizialmente allineate le scansioni laser utilizzando l'informazione di posizione più precisa per ogni dataset, GPS e VO rispettivamente per KITTI (e KITTI-360) e Oxford Robotcar Radar. Applicando solo meccanismi di allineamento si è notato come il risultato non fosse sufficiente per lo scopo della tesi e di conseguenza, si è deciso di applicare algoritmi di allineamento locale come *Iterative Closest Point* (ICP) [2][6][47] per allineare con più precisione una scansione rispetto ad un'altra. In particolare, è stato utilizzato Generalized ICP (G-ICP), variante di ICP sviluppata da Segal, Haehnel e Thrun [40], in grado di ottenere un risultato molto più accurato rispetto a ICP (ad esempio allineando in maniera molto precisa i singoli punti dei muri), ma a discapito di un aumento dei tempi di computazione. Ottenere il valore di ground truth è fondamentale per effettuare una validazione corretta della stima di profondità e, di conseguenza, si è deciso di dare priorità all'accuratezza anche a discapito di un maggiore tempo di computazione. Questo non è un problema in quanto il task di allineamento e registrazione avviene offline.

6. Utilizzando la proiezione centrale della camera, si è in grado di poter proiettare i punti 3D in una immagine 2D. In questa fase, ad ogni punto 2D viene asso-

ciato il relativo valore di profondità. Inoltre, se in un pixel vengono proiettati due o più punti, viene tenuto il punto con valore di profondità inferiore. Ciò che si ottiene al termine di questa fase è un matrice monodimensionale le cui dimensioni sono pari a quelle dell'immagine. Se in una cella (pixel) è contenuto almeno un punto viene impostato il valore di profondità più vicino, altrimenti se non ci sono punti che ricadono in quel pixel il valore viene impostato a 0.

7. Effettuando la registrazione di numerose pointcloud, la proiezione dei punti sul piano immagine può produrre una mappa di profondità con un alto grado di rumorosità. Infatti, è possibile che vengano proiettati punti occlusi rispetto al punto di vista dell'osservatore camera (ad esempio punti dietro i muri o altri ostacoli) in quanto le singole scansioni sono prese ad istanti temporali diversi. Per risolvere questo problema viene adottato il filtro di stima delle occlusioni presentato da Pintus, Gobbetti e Agus [36] in grado di determinare i punti occlusi costruendo per ogni punto un cono sulla linea di proiezione che porta al centro immagine. Se nell'apertura del cono, definita da un parametro *threshold*, sono presenti dei punti allora il punto considerato è occluso, altrimenti viene considerato visibile.

Nella fase finale del progetto viene effettuata la valutazione dell'errore confrontando la mappa di profondità predetta da una rete neurale con la mappa LIDAR ottenuta nelle fasi precedenti. Con l'obiettivo di valutare e confrontare le performance di varie reti che effettuano stima di profondità, in questo lavoro viene utilizzato il metodo di valutazione proposto da Eigen, Puhrsch e Fergus [10], comunemente utilizzato in letteratura per questo task, che definisce cinque indicatori per valutare la bontà della predizione: RMSE, RMSE log, Abs Rel, Sq Rel e Accuracies; e limita la massima profondità stimabile a 80 metri.
Dai risultati ottenuti si può concludere che dopo aver applicato l'approccio sviluppato si è notata una generale variazione delle performance delle diverse reti utilizzate. In particolare, anche se le reti ottengono un buon errore con punti vicini, all'aumentare della distanza cresce anche l'errore di stima di profondità dei punti e, poichè, sono stati confrontati tanti punti a grandi distanze, in generale si ottiene un aumento dell'errore medio. Questo progetto può costituire un buon di punto di partenza per stimare con più precisione modelli neurali sia supervisionati che non supervisionati. Infine, si potrebbe pensare di utilizzare le coppie immagine e mappe di profondità dense direttamente in fase di training per reti neurali supervisionate con l'idea di migliorarne le performance.
Il lavoro viene presentato articolandolo come segue: nel Capitolo 2 viene presentato lo stato dell'arte riguardante la ricostruzione 3D tramite camere, le tecniche basate

su reti neurali e una breve descrizione dei possibili metodi di rappresentazioni dell'ambiente e dei più comuni dataset in letteratura in quest'ambito; nel Capitolo 3 vengono presentate le motivazioni che hanno portato alla scelta di sviluppare questo lavoro, mostrando la pipeline di valutazione e i dataset utilizzati; nel Capitolo 4 viene illustrata la configurazione degli esperimenti mostrando il protocollo di valutazione, le reti neurali e i loro risultati dei test; infine nel Capitolo 5 vengono tratte le conclusioni del lavoro di tesi.

Capitolo 2

Stato dell'arte

In questo capitolo si analizza lo stato dell'arte rispetto al problema della stima di profondità tramite camere. Nella prima parte viene effettuata una introduzione sull'argomento spiegando i problemi della ricostruzione 3D e i vari metodi che possono essere applicati per ottenere l'informazione di profondità. Successivamente viene mostrato il principio della visione stereoscopica, con i processi di eliminazione delle distorsioni e rettificazione. Vengono trattati i due problemi principali di un sistema stereoscopico: la ricerca di corrispondenze e ricostruzione 3D, mostrando vantaggi e svantaggi di avere una linea di base più o meno lunga. La parte successiva presenta le diverse reti neurali supervisionate e non supervisionate per stimare la profondità di una scena. Vengono descritte le funzioni di loss e definite le metriche per la valutazione degli errori utilizzate in letteratura. In seguito viene introdotto il concetto di mappa e vengono mostrati alcuni modelli usati in letteratura per rappresentare lo spazio operativo del robot. Infine, vengono introdotti alcuni tra i principali dataset presenti in letteratura che vengono utilizzati anche come benchmark per la stima di profondità.

2.1 Problema della ricostruzione 3D

La stima di profondità da immagini è uno degli obiettivi fondamentali della computer vision, in quanto sistemi autonomi utilizzano l'informazione di profondità per percepire ostacoli o localizzarsi rispetto ad una mappa permettendo di portare a compimento task come *Simultaneous Localization and Mapping* (SLAM), navigazione e riconoscimento di oggetti. Le tecniche per ottenere questo dato si dividono in tre grandi famiglie: metodi basati sulla geometria, metodi basati sui sensori e metodi basati su deep learning.

Metodi basati sulla geometria

I metodi che si basano sulla geometria recuperano la struttura 3D da una coppia di immagini utilizzando vincoli geometrici. *Structure from motion* (SfM) [44] è uno dei principali metodi per la stima della struttura tridimensionale dell'ambiente a partire da sequenze di immagini 2D. Viene comunemente applicato a compiti di ricostruzione 3D e SLAM. La profondità delle feature sparse può essere gestita da SfM attraverso le corrispondenze di feature e vincoli geometrici tra le sequenze di immagini. Di conseguenza, l'accuratezza della stima della profondità si basa fortemente sulla corretta associazione delle feature e su sequenze di immagini in alta risoluzione. Inoltre, utilizzando sequenze monoculari non viene risolto il problema della scala globale della scena. Analogamente, la visione stereoscopica permette di ricostruire la struttura tridimensionale osservando la scena da due punti di vista. Conoscendo in anticipo la trasformazione tra le camere, vengono calcolate le mappe di disparità e il dato di scala può essere ricavato dalla stima di profondità. In generale, i metodi che si basano sulla geometria fanno affidamento a coppie di immagini o sequenze per ottenere il dato di profondità in quanto strettamente correlati alla soluzione geometrica del problema, pertanto non è possibile ottenere una mappa di profondità da singola immagine con un discreto grado di accuratezza.

Metodi basati sui sensori

Altri metodi si basano su sensori come possono essere camere RGB-D e laser scanner per acquisire il dato di profondità. Il vantaggio di questi sensori è dato dal fatto che sono in grado di ottenere direttamente l'informazione di profondità. Le camere RGB-D acquisiscono, oltre al dato RGB, una immagine di profondità della scena, ottenendo mappe dense, dove ad ogni pixel immagine è associato il relativo valore di profondità. La loro applicazione si limita ad ambienti indoor in quanto hanno un campo di misurazione limitato che non permette il loro utilizzo outdoor. I laser scanner, chiamati anche *Light Detection and Ranging* (LIDAR), permettono di determinare la distanza di un oggetto o di una superficie utilizzando un impulso luminoso. Ad oggi, pur nella loro non perfezione, in quanto la mappa di profondità generata è sparsa, rappresentano lo stato dell'arte per quanto riguarda la precisione nelle misurazioni e l'affidabilità in condizioni diverse. La robustezza del laser deriva dal fatto che lavora solo ad una frequenza ed emette un fascio luminoso che arriva concentrato su un target, generando un eco intenso anche a distanze elevate e su materiali con riflettività non tanto alta. Tuttavia, in ambito automotive, a causa degli svantaggi legati al costo elevato e alla presenza di una componente meccanica

che li rende poco affidabili e durevoli nelle condizioni di lavoro di un autoveicolo, l'adozione su grande scala risulta molto difficoltosa. Di conseguenza, grazie a bassi costi, a un notevole campo di misurazione e ad un già presente utilizzo in una grande varietà di altre applicazioni, per ricavare l'informazione di profondità vengono adottate camere monoculari o in configurazione stereo.

Metodi basati su deep learning

Il rapido sviluppo negli ultimi anni del deep learning ha portato anche allo studio di *Deep Neural Networks* (DNN) in grado di effettuare task di image processing ottenendo risultati migliori dei metodi tradizionali presenti in letteratura: come classificazione [46], riconoscimento di oggetti [34] e segmentazione semantica [29]. La ricerca in questo settore ha portato allo studio di DNN in grado di stimare la profondità da singole immagini. Nel corso del tempo sono stati proposti diversi approcci che hanno permesso di migliorare l'accuratezza delle predizioni fino ad ottenere performance promettenti anche in contesti molto complicati come la ricostruzione di scene urbane dinamiche. Gli approcci supervisionati si basano sulla esistenza di dataset contenenti coppie di immagini e dato di profondità per supervisionare in fase di training la stima di profondità. Tuttavia, poiché è difficile acquisire grandi quantità di esempi etichettati con il dato di profondità, in letteratura sono emersi approcci alternativi che si basano sull'apprendimento non supervisionato, con il vantaggio di poter utilizzare un insieme ristretto di esempi etichettati solo per la fase di valutazione.

2.2 Stereo-visione

In questa sezione viene descritta la tecnica di visione stereoscopica che permette, utilizzando una coppia di camere, di ottenere un sistema di visione in grado di stimare la profondità degli oggetti nella scena osservata.

2.2.1 Fondamenti della stereo-visione

Per stereo-visione ci si riferisce all'abilità di inferire informazioni sulla struttura tridimensionale e della profondità di una scena a partire da due o più immagini prese da differenti punti di vista.
La geometria di un sistema stereoscopico, conosciuta come *geometria epipolare*, viene mostrata in figura 2.1. Utilizzando il modello pinhole per rappresentare ciascuna camera, è possibile definire i centri di proiezione O_l e O_r, i piani immagine π_l e π_r e la

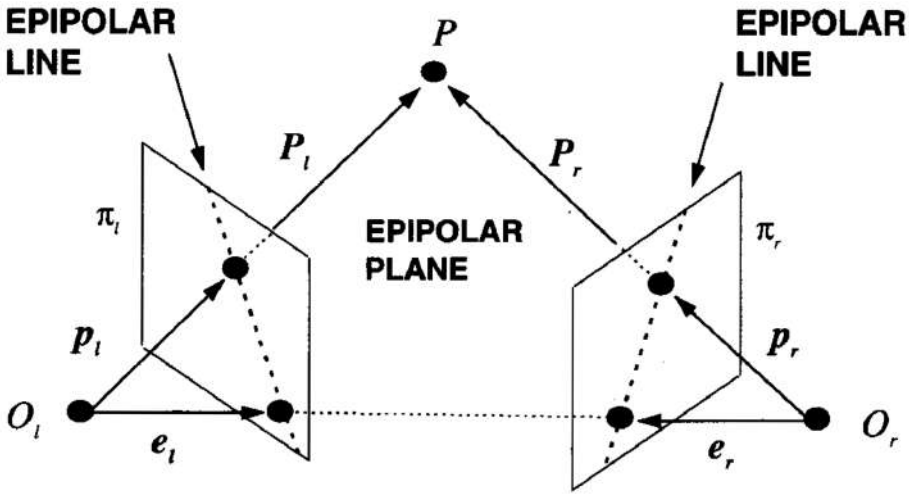


Figura 2.1: Geometria Epipolare. Immagine presa da [43].

linea di base che collega i due centri di proiezione definendo, con la sua intersezione sul piano immagine, gli *epipoli*, rispettivamente e_l e e_r. Le lunghezze focali sono denotate da f_l e f_r.

Ogni camera identifica un sistema di riferimento la cui origine è il centro di proiezione e l'asse z è rappresentato dall'asse ottico. I due sistemi di riferimento sono in relazione tramite i parametri estrinseci che definiscono una trasformazione rigida nello spazio 3D, caratterizzata da un vettore di traslazione $T = O_r - O_l$ e una matrice di rotazione R. Dato un punto P nello spazio, la relazione tra le sue proiezioni P_l e P_r sui piani immagine è la seguente:

$$P_r = R(P_l - T) \tag{2.1}$$

La relazione tra un punto nello spazio 3D e la sua proiezione è descritta dall'equazione della proiezione prospettica:

$$p_l = \frac{f_l}{Z_l} P_l \tag{2.2}$$

$$p_r = \frac{f_r}{Z_r} P_r \tag{2.3}$$

Viene definito *piano epipolare* il piano identificato da P, O_l e O_r. L'intersezione tra il piano epipolare e il piano immagine definisce la *linea epipolare*.
L'importanza della geometria epipolare è rappresentata dal fatto che grazie a un vincolo, detto *vincolo epipolare*, viene limitata la ricerca della corrispondenza dei punti tra le immagini ad una sola retta. Infatti, grazie a questo vincolo, nel caso si voglia identificare il punto p_r, corrispondente di p_l, è possibile ridurre lo spazio di

ricerca delle corrispondenze all'intersezione tra il piano epipolare e il piano immagine, ovvero, la linea epipolare.

2.2.2 Eliminazione delle distorsioni

La qualità dell'immagine acquisita da parte di un sensore camera dipende dalla capacità della sua ottica di concentrare il segnale luminoso sugli elementi sensibili. Tuttavia, l'utilizzo di lenti nell'ottica introduce *aberrazioni* e *distorsioni* che introducono artefatti nell'acquisizione delle immagini. In particolare, i principali tipi di distorsione vengono chiamati *a barilotto* e *a cuscinetto* e la loro intensità è determinata dalla qualità dell'ottica utilizzata. Queste distorsioni di tipo radiale possono essere corrette utilizzando il modello di distorsione di Brown[3], conosciuto anche come modello di Brown-Conrady basato sul precendente lavoro di Conrady[7]. Il modello di Brown-Conrady è in grado di correggere sia le distorsioni (in inglese operazione di *undistort*) radiali che quelle tangenziali causate dai componenti della lente non perfettamente allineati.
Il modello di Brown-Conrady è descritto di seguito:

$$\begin{aligned} x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + \cdots) + \\ (P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c))(1 + P_3 r^2 + P_4 r^4 + \cdots) \end{aligned} \tag{2.4}$$

$$\begin{aligned} y_u = y_d + (y_d - y_c)(K_1 r^2 + K_2 r^4 + \cdots) + \\ (2P_1(x_d - x_c)(y_d - y_c) + P_2(r^2 + 2(y_d - y_c)^2))(1 + P_3 r^2 + P_4 r^4 + \cdots) \end{aligned} \tag{2.5}$$

dove:

- (x_d, y_d) sono i punti immagine distorti proiettati sul piano immagine usando una specifica lente;
- (x_u, y_u) sono i punti immagine senza distorsione proiettati secondo il modello camera a pinhole;
- (x_c, y_c) è il centro di distorsione;
- K_n è l'n-esimo coefficiente radiale di distorsione;
- P_n è l'n-esimo coefficiente tangenziale di distorsione;
- $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$ è la distanza euclidea tra i punti distorti e il centro di distorsione.

2.2.3 Rettificazione

Prima di effettuare qualsiasi operazione di ricerca di corrispondenze è necessario definire una trasformazione dei piani immagine in modo tale che le coppie di linee epipolari coniugate diventino parallele ad uno degli assi del piano immagine. Questa operazione viene definita *rettificazione*. In figura 2.2 viene mostrato un esempio con una coppia di immagini stereoscopiche.
In letteratura gli algoritmi di rettificazione vengono classificati in tre categorie: rettificazione planare [12], rettificazione cilindrica [33] e rettificazione polare [37].
In generale, al termine del processo di rettificazione, devono valere sempre due proprietà:

- le linee epipolari associate a un punto P nello spazio 3D sono parallele all'asse orizzontale;
- punti corrispondenti nel 3D hanno identiche coordinate verticali sul piano immagine.

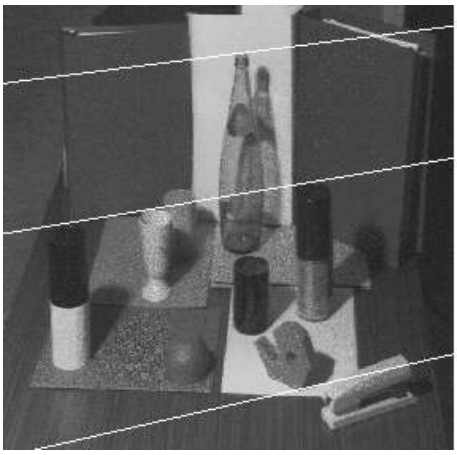
L'intera procedura di rimozione delle distorsioni e rettificazione viene mostrata in figura 2.3. Le diverse fasi possono così essere riassunte: acquisizione delle immagini, rimozione della distorsione, rettificazione e ritaglio delle immagini rettificate.

2.2.4 Disparità e ricostruzione 3D

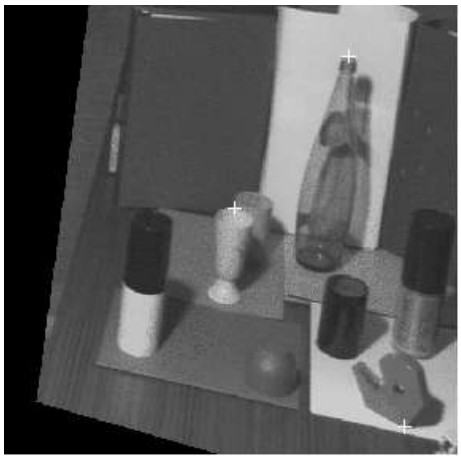
La rimozione delle distorsioni e la rettificazione permettono di portare a compimento due task: la ricerca delle corrispondenze tra due immagini e successivamente la ricostruzione 3D della scena. Il primo, conosciuto come *problema delle corrispondenze* consiste nel determinare la corrispondenza tra elementi immagine nelle due camere. In particolare, non tutti gli elementi immagine osservati da una camera hanno delle corrispondenze, in quanto alcune parti della scena potrebbero non essere visibili dal secondo punto di vista. Di conseguenza, un sistema stereoscopico deve anche determinare quali parti dell'immagine non devono essere associate. Quindi, una volta conosciute le corrispondenze, è possibile affrontare il problema della *ricostruzione*. Le corrispondenze di punti sui piani immagine definiscono una mappa, denominata *mappa di disparità*. Se la geometria del sistema stereoscopico è nota, la mappa di disparità può essere convertita in una mappa 3D della scena. Questo procedimento è detto *ricostruzione*.

In figura 2.4 viene rappresentato un sistema stereoscopico con due camere, visto dall'alto. I piani immagine sinistro e destro sono complanari, rappresentati rispettivamente dai segmenti I_l e I_r. O_l e O_r sono i centri di proiezione. La posizione nello

(a) Immagine di sinistra



(b) Immagine di destra



(c) Immagine di sinistra rettificata

(d) Immagine di destra rettificata

Figura 2.2: Coppia di immagini stereo prima e dopo la rettificazione. Nell'immagine di destra vengono mostrate le linee epipolari corrispondenti ai punti identificati nell'immagine di sinistra. Si può notare come, in generale, l'immagine rettificata non sia contenuta nella stessa regione del piano immagine dell'immagine originale. Immagine presa da [43].

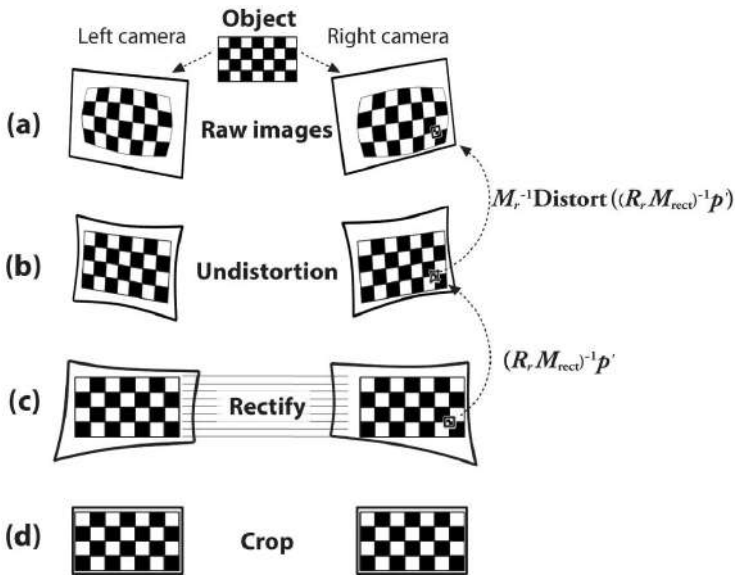
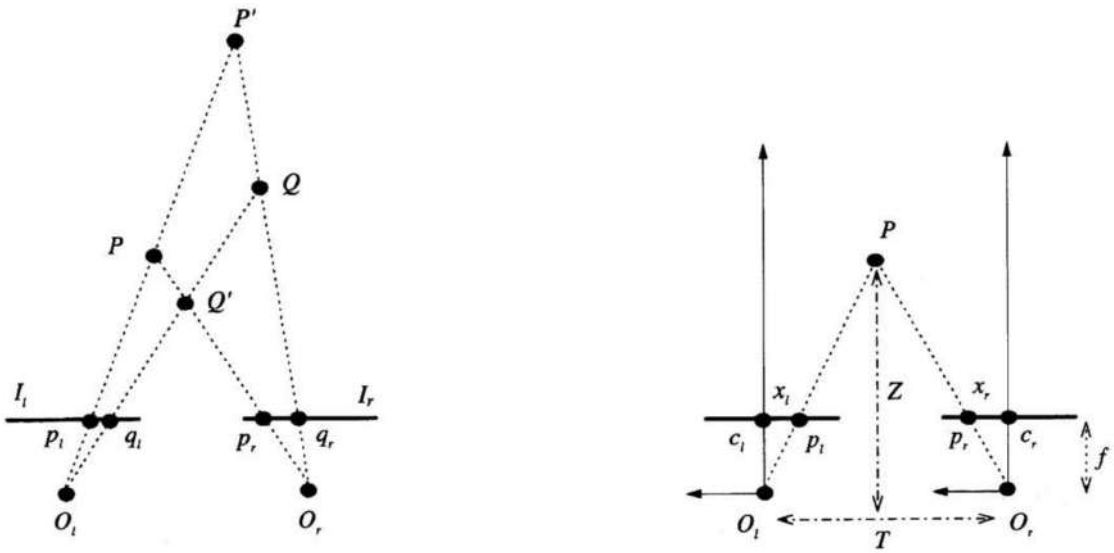


Figura 2.3: Processo di undistort e rettificazione delle immagini stereo.



(a) (b)

Figura 2.4: Sistema stereoscopico. (a) La ricostruzione 3D dipende dalla soluzione del problema di corrispondenze. (b) La profondità è stimata dalla disparità dei punti corrispondenti. Immagine presa da [43].

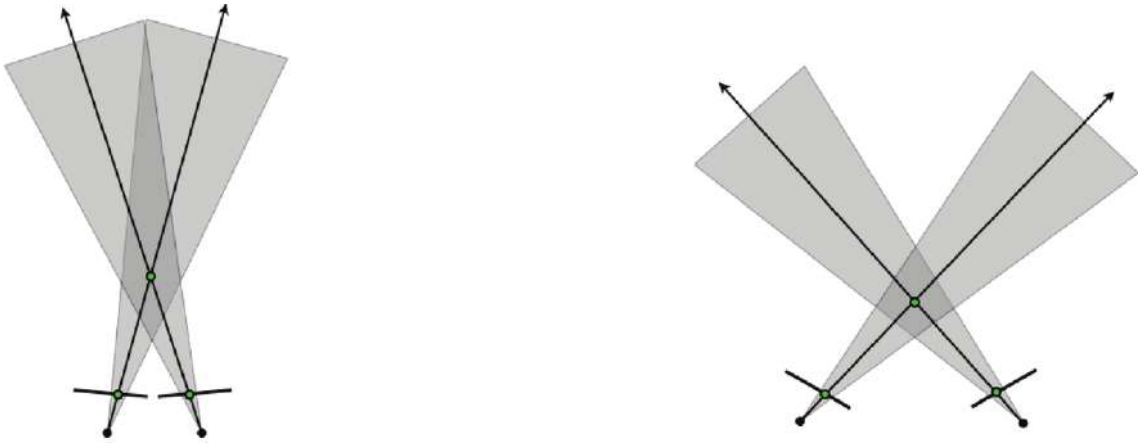
spazio dei punti P e Q viene determinata tramite la tecnica chiamata *triangolazione*. In particolare, questa tecnica sfrutta le intersezioni dei raggi definiti dai centri di proiezione e le immagini di P e Q, rispettivamente p_l e q_l per il piano immagine di sinistra e p_r e q_r per quello di destra. La tecnica di triangolazione dipende strettamente dalla soluzione del problema delle corrispondenze. Infatti, se le corrispondenze trovate non sono accurate, il risultato che si ottiene sarà una ricostruzione di bassa qualità. Grazie al vincolo epipolare, è possibile limitare lo spazio di ricerca delle corrispondenze e ridurre la complessità del problema da quadratica a lineare. Quindi, assumendo che sia possibile risolvere il problema delle corrispondenze, ci si può concentrare sul problema della ricostruzione della posizione del punto P nello spazio 3D, considerando le sue proiezioni p_l e p_r sul piano immagine. Siano x_l e x_r le coordinate di p_l e p_r rispetto ai punti principali c_l e c_r. Sia f la lunghezza focale e Z la distanza tra P e la linea di base. Effettuando la congruenza tra triangoli simili (p_l, P, p_r) e (O_l, P, O_r) si ha:

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z} \tag{2.6}$$

e risolvendo 2.6 rispetto a Z si ottiene:

$$Z = f\frac{T}{d} \tag{2.7}$$

dove $d = x_r - x_l$ è la *disparità* e misura la differenza di posizione tra i punti corrispondenti nelle due immagini. Dall'equazione 2.7 si può vedere come la profondità è inversamente proporzionale alla disparità.
Per risolvere il problema della ricostruzione la scelta della linea di base risulta importante. Al variare della lunghezza della linea di base varia la qualità della ricostruzione che viene effettuata. Infatti, come si può notare in figura 2.5, una più corta linea di base porta ad avere una incertezza maggiore sulla distanza dei punti proiettati in entrambi i piani immagine. Dall'altra parte, una linea di base più lunga permette di ottenere una stima migliore ma a discapito di una maggiore difficoltà nella ricerca delle corrispondenze. Infatti, la presenza di punti occlusi o la diversa intensità di luce e colore acquisite dal sensore sono alcuni dei problemi che rendono difficile effettuare la corrispondenza dei punti. Con una linea di base corta questi problemi vengono mitigati in quanto le camere hanno una visione della scena da prospettive simili e di conseguenza l'intensità della luce non dovrebbe variare di molto. È necessario, quindi, trovare un compromesso tra la precisione della ricostruzione e ricerca di corrispondenze per definire la giusta lunghezza della linea di base per lo scopo che si vuole raggiungere.



(a) Linea di base corta.

(b) Linea di base lunga.

Figura 2.5: Differenze tra diverse lunghezze della linea di base. Una linea di base grande comporta una maggiore stima della profondità ma a discapito di una difficoltà nella ricerca delle corrispondenze.

2.3 Ricostruzione 3D mediante reti neurali

Come descritto nelle sezioni precedenti, la stima di profondità ha un ruolo fondamentale nel ricostruire la geometria della scena. In letteratura sono presenti una grande quantità di lavori sulla stima di profondità a partire da immagini stereo o dal movimento [39]. Utilizzando un sistema stereo, la profondità può essere ottenuta in modo deterministico a patto che si forniscano delle corrispondenze immagine accurate. Per quanto riguarda invece la stima di profondità a partire da una singola immagine è necessario trovare al suo interno indizi relativi alla profondità come possono essere la prospettiva, le dimensioni degli oggetti e la posizione della camera. Inoltre, potrebbe essere necessaria una vista globale della scena per poter correlare questi indizi in modo efficace (ad esempio oggetti in scala possono sembrare più grandi in base alla prospettiva e quindi falsano la stima di profondità). Del resto, anche combinando queste informazioni non è detto che si riesca ad ottenere una stima accurata. Infatti, data un'immagine ci possono essere un numero infinito di scene reali che potrebbero averla generata. Riducendo lo spazio di ricerca, il dato di profondità rimane comunque legato al valore di scala globale della scena. Riuscire a calcolare un valore di scala simile al vero permetterebbe di ottenere una maggiore accuratezza nella ricostruzione 3D della scena. Un approccio standard con le reti neurali è quello di fornire un valore reale di disparità o profondità durante la fase di training. Tuttavia, una problematica attuale dei dataset in ambito automotive è la mancanza di valori di ground truth in quanto sono molto difficili da acquisire. Di conseguenza sono nati approcci che usano tecniche di apprendimento non supervi-

sionato. In questa sezione vengono esaminati alcuni approcci basati su reti neurali che effettuano la stima di profondità, mostrando diverse tipologie di apprendimento automatico: metodi supervisionati e non supervisionati. Anche se i processi di training dei metodi non supervisionati fanno affidamento a video monoculari o coppie di immagini stereo, in fase di inferenza i modelli di rete neurale effettuano la predizione delle mappe di profondità da una singola immagine. Inoltre, nella parte conclusiva vengono mostrate le metriche comunemente utilizzate in letteratura per valutare l'errore di stima di profondità.

2.3.1 Apprendimento supervisionato

Una rete con apprendimento supervisionato utilizza un dataset con esempi etichettati, coppie input-output, dove l'output rappresenta il valore reale desiderato, per allenare un modello rispetto ad uno specifico task (ad esempio classificazione, regressione, etc.).

Modello base per metodi supervsionati

Nelle DNN (Deep Neural Networks) per la stima della profondità, l'apprendimento supervisionato sfrutta delle immagini ground truth di profondità ed il task assume le caratteristiche di un problema di regressione. Lo scopo delle reti neurali è quindi quello di predire le mappe di profondità da singola immagine. Le differenze tra la profondità stimata e il valore reale sono sfruttate per supervisionare la fase di training delle reti utilizzando la funzione di loss $\mathcal{L}_2$:

$$\mathcal{L}_2(d, d^*) = \frac{1}{N} \sum_{i}^{N} ||d - d^*||_2^2 \tag{2.8}$$

dove d è il valore predetto e d^* è il valore ground truth. Pertanto, le reti apprendono l'informazione di profondità della scena approssimando il valore reale.

Differenti architetture e funzioni di loss di metodi supervisionati

Il lavoro di Eigen, Puhrsch e Fergus [10] è uno dei primi in letteratura che affronta il problema di stima di profondità a partire da una singola immagine con reti neurali convoluzionali (CNNs). L'architettura proposta, mostrata in figura 2.6, è formata da due componenti: il primo, *global coarse-scale network*, stima la struttura globale della scena, mentre il secondo, *local fine-scale network*, la rifinisce sfruttando informazioni locali. Lo scopo del global coarse-scale network, quindi, è quello di predire la struttura complessiva della mappa di profondità usando l'informazione

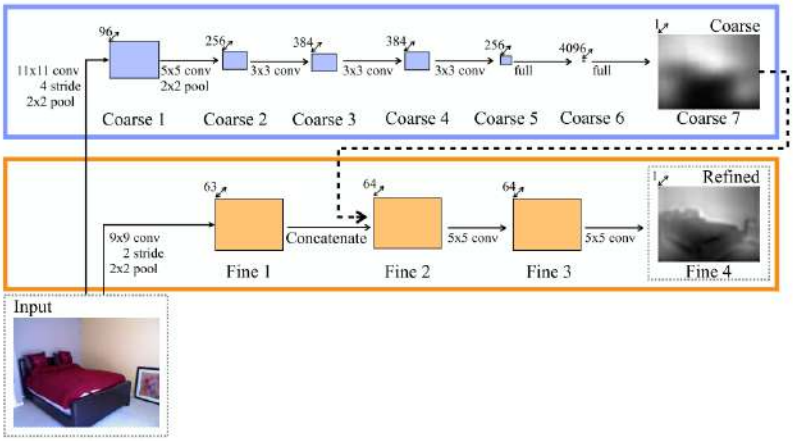


Figura 2.6: Architettura di rete. Immagine presa da [10].

dall'intera immagine. Gli strati superiori della rete contengono l'intera immagine nel campo visivo, mentre quelli centrali e inferiori combinano informazioni da diverse parti dell'immagine tramite operazioni di max-pooling. Così facendo, la rete è in grado di integrare l'informazione globale dell'intera scena per stimare la profondità. Tale comprensione è necessaria, nel caso di una singola immagine, per utilizzare in modo efficiente indizi nella scena, come la posizione degli oggetti o l'allineamento della stanza, in quanto una vista locale non è in grado di notare queste importanti feature. Successivamente, la comprensione della rete rispetto ai dettagli locali viene rifinita utilizzando il local fine-scale network. Lo scopo di questo componente è di modificare la stima grossolana allineando i dettagli locali come oggetti e bordi dei muri. L'output del global coarse-scale network viene usato come input nel local fine-scale network così che la predizione globale venga modificata per incorporare dettagli più precisi. Eigen, Puhrsch e Fergus hanno dimostrato come l'utilizzo di un valore impreciso di scala globale della scena rappresenta una grande parte dell'errore totale, in quanto il valore medio della stima di profondità influisce molto sull'errore totale. Di conseguenza, viene utilizzato un errore di scala invariante che misura la relazione tra i punti nella scena senza tenere conte della scala globale assoluta. Si definisce quindi la *scale-invariant mean squared error loss* come:

$$L(y, y^*) = \frac{1}{N}\sum_i (\log y_i - \log y_i^*)^2 - \frac{\lambda}{n^2}(\sum_i (\log y_i - \log y_i^*))^2 \tag{2.9}$$

dove:

- y è la mappa di profondità stimata;
- y^* è la mappa di profondità ground truth;
- $\lambda \in [0, 1]$ è un fattore di bilanciamento (di norma ha valore 0.5).

L'output della rete è $\log y$ ovvero la stima di profondità logaritmica data dall'ultimo strato. Durante la fase di training se sono presenti valori mancanti nella mappa di profondità, per esempio in prossimità di confini di oggetti o superfici riflettenti, la loss valuta solo i punti esistenti modificando il valore di n nell'equazione 2.9. Eigen e Fergus [9] propongono un framework multi-scala in grado di affrontare il problema di stima di profondità, stima della normale di superfici e predizione di etichette semantiche da singola immagine. Partendo dall'equazione 2.9, gli autori propongono la seguente funzione di loss per favorire la consistenza strutturale locale:

$$\mathcal{L}_s = \frac{1}{N}\sum_i n[(\nabla_x D_i)^2 + (\nabla_y D_i)^2] \tag{2.10}$$

dove $D_i = \log(d_i) - \log(d_i^*)$ e ∇ è il vettore operatore differenziale. La funzione calcola i gradienti rispetto alle direzioni orizzontali e verticali dell'immagine tenendo conto della differenza tra la profondità predetta e la ground truth. L'optical flow, ovvero il movimento degli oggetti rispetto all'osservatore, è quindi risolto usando la CNN supervisionata. Mayer et al. [32] propongono un lavoro che effettua optical flow e stima di disparità. Ispirati da ResNet [18], Laina et al. [23] utilizzano una tecnica chiamata *residual learning*, in grado di saltare dei layer nella rete, per permettere ad una rete di apprendere le relazioni tra le mappe di profondità e le immagini. Durante il processo di training, come funzione di loss viene usata la reverse Huber (Berhu) [50]:

$$\mathcal{L}_{Berhu}(d, d^*) = \begin{cases} |d - d^*|, & \text{se} |d - d^*| \leq c, \\ \frac{(d-d^*)^2 + c^2}{2c}, & \text{se} |d - d^*| > c, \end{cases} \tag{2.11}$$

dove c rappresenta il valore limite impostato a $\frac{1}{5} max_i(|d - d^+|)$. Se $|x| < c$, allora la funzione di loss Berhu assume lo stesso significato della distanza euclidea $\mathcal{L}_1$, altrimenti sarebbe uguale a $\mathcal{L}_2$. Grazie a questa funzione di loss e a una rete più profonda questo lavoro ottiene risultati migliori dei precedenti.
Come si è visto in Eigen, Puhrsch e Fergus [10] il problema di stima di profondità monoculare viene formulato come un problema di regressione e, anche se l'ottimizzazione può essere ottenuta con una soluzione ragionevole, questo comporta però una convergenza lenta il cui risultato finale si discosta molto dalla ground truth. Di conseguenza, per ovviare tali problemi, Fu et al. [11] riformulano il problema di apprendimento della rete come un problema di regressione ordinale. L'architettura della rete proposta, mostrata in figura 2.7, si suddivide in due componenti principali chiamate rispettivamente *dense feature extractor* e *scene understanding modular*.

Il primo estrae feature dense incorporando informazione multi-scala. Il secondo è formato da moduli paralleli in grado di estrarre feature e catturare l'informazio-

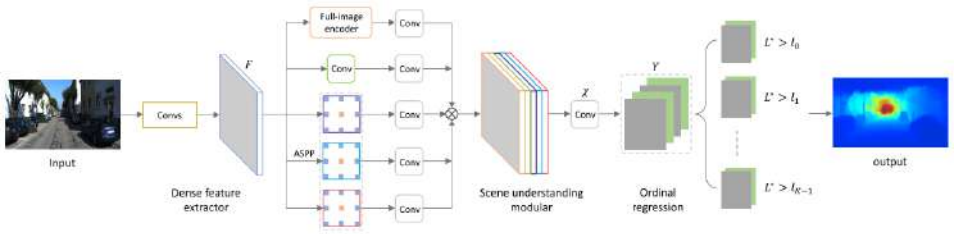


Figura 2.7: Architettura di rete. Immagine presa da [11].

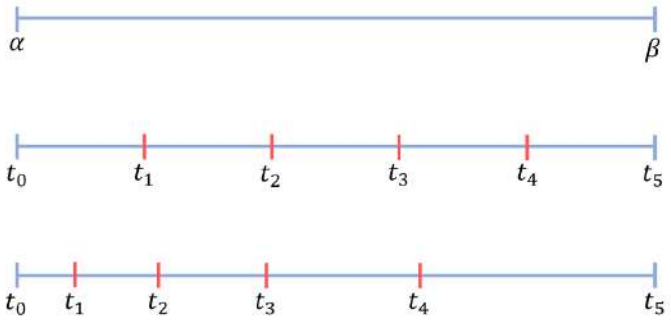


Figura 2.8: Intervalli di discretizzazione: UD al centro e SID in bassso. Immagine presa da [11].

ne globale per ridurre l'errore nella stima di profondità a livello locale. Le feature ottenute da questi moduli vengono concatenate per ottenere una rappresentazione completa dell'immagine di input. Infine, le feature map rappresentanti diversi livelli di dettaglio (globale e locale) vengono concatenate.
Per quantizzare un intervallo di profondità $[\alpha, \beta]$ in un insieme di valori discreti viene usata una tecnica *spacing-increasing discretization* (SID) rispetto ad una discretizzazione uniforme (UD) che non discrimina i valori all'interno di un intervallo. Come si vede in figura 2.8, la tecnica SID considera un errore di stima crescente all'aumentare della distanza. In questo modo, è possibile ridurre la loss in regioni con più grandi valori di profondità così che la rete sia in grado di predire in modo più accurato valori di profondità piccoli e medi e stimare valori grandi con meno errore.

La formula di SID è la seguente:

$$t_i = e^{\log(\alpha) + \frac{\log(\beta/\alpha) i}{K}} \tag{2.12}$$

dove $t_i \in \{t_0, t_1, \ldots, t_K\}$ sono le soglie di discretizzazione. Successivamente, il problema di stima della profondità viene trasformato in un problema di regressione ordinale, adottando *softmax* come funzione di loss per apprendere i parametri. Viene, inoltre, definita la funzione di loss ordinale $\mathcal{L}(\chi, \Theta)$ come la loss media per pixel

$\Psi(h, w, \chi, \Theta)$ sull'intero dominio dell'immagine:

$$\mathcal{L}(\chi, \Theta) = -\frac{1}{N} \sum_{w=0}^{W-1} \sum_{h=0}^{H-1} \Psi(w, h, \chi, \Theta) \tag{2.13}$$

e

$$\Psi(h, w, \chi, \Theta) = \sum_{k=0}^{l_{(w,h)-1}} \log(\mathcal{P}^K_{(w,h)}) + \sum_{k=l_{(w,h)}}^{K-1} (1 - \log(\mathcal{P}^k_{w,h})) \tag{2.14}$$

$$\mathcal{P}^k_{(w,h)} = P(\hat{l}_{(w,h)} > k | \chi, \Theta) \tag{2.15}$$

dove:

- $\chi = \varphi(I, \Phi)$ sono le mappe di features di dimensione W x H x C data una immagine I e Φ sono i parametri delle parti della rete;
- $\Theta = (\theta_0, \theta_1, \theta_2, \dots)$ contiene i vettori dei pesi;
- $l_{(w,h)} \in 0, 1, \dots, K-1$ è l'etichetta discreta prodotta da SID alla locazione spaziale (w, h);
- $\mathcal{N} = W \times H$;
- $\hat{l}_{(w,h)}$ è il valore stimato discreto di decodifica da $y_{(w,h)}$.

Minimizzare 2.13 garantisce che predizioni più lontane dall'etichetta subiscano una penalità maggiore rispetto a quelle più vicine, in quanto sono più distanti dal valore reale.

Metodi basati su conditional random fields (CRF)

Nel lavoro di Li et al. [24] viene formulata la stima di profondità come un problema di apprendimento basato su *deep continuous Conditional Random Fields* (CRF) [22], senza affidarsi a conoscenze a priori della geometria della scena o altro, dove CRF sono una classe di metodi di modellazione statistica a grafo per effettuare predizioni strutturate. Considerando che i valori di profondità sono continui, CRF può rifinire i valori di profondità considerando l'informazione di pixel adiacenti. Avendo come scopo l'inferenza di ogni pixel nell'immagine, si assume che l'immagine sia formata da regioni omogenee (superpixel) e si considera il modello a grafo CRF composto da nodi che rappresentano queste regioni. Viene definita una funzione di energia **E(d)** formata da tre parti: la prima calcola la distanza quadratica tra il valore di profondità vero e quello predetto, la seconda rappresenta lo *smoothing* per forzare la rilevanza tra superpixel vicini e la terza rappresenta un modello auto regressivo per

descrivere la struttura locale nella mappa di profondità. Liu et al. [27] propongono un lavoro simile che usa una metodologia di *pooling*, che discretizza i superpixel, per migliorare l'accuratezza della stima.

2.3.2 Apprendimento non supervisionato

Una rete neurale non supervisionata apprende dei pattern da dati non etichettati e impara a costruire una rappresentazione interna del dataset di training. Lavori che effettuano la stima di profondità *self-supervised* considerano l'apprendimento come un problema di *novel view synthesis*, addestrando una rete per predire l'aspetto di un'immagine target dal punto di vista di un'altra immagine. In alcuni lavori, come ad esempio [14], si assume che la scena sia statica con solo il movimento della camera, ma quando questa assunzione non è più vera, come in situazioni di traffico urbano dove la scena è molto dinamica, le prestazioni possono peggiorare molto, come ad esempio in [28]. Questa tipologia di reti sono il focus centrale della tesi e di seguito vengono descritti diversi lavori che, con i loro approcci, cercano di superare i problemi relativi alla stima di profondità da singola immagine.

Modello base per metodi monoculari non supervisionati

I metodi non supervisionati sono allenati utilizzando sequenze di immagini monoculari e i vincoli geometrici sono costruiti sulla proiezione tra frame vicini:

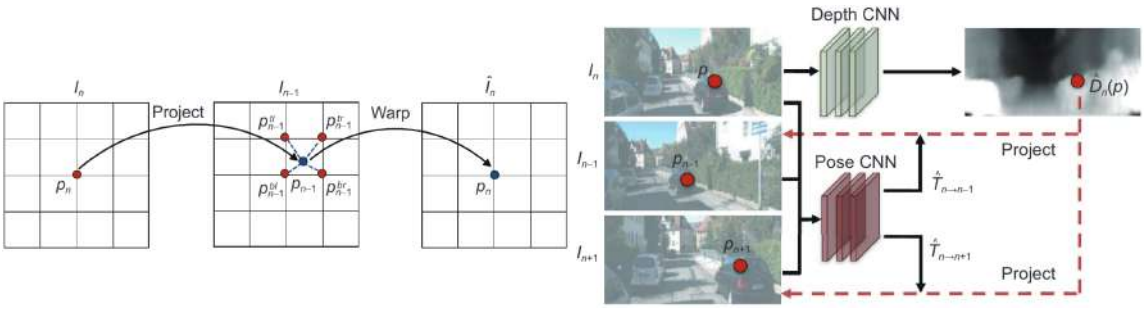
$$p_{n-1} \sim K T_{n \to n-1} D_n(p_n) K^{-1} p_n \tag{2.16}$$

dove:

- p_n sono i pixel dell'immagine I_n;
- p_{n-1} sono i corrispondenti pixel p_n nell'immagine I_{n-1};
- K è la matrice dei parametri intrinseci della camera (conosciuta);
- $D_n(p_n)$ denota il valore di profondità del pixel p_n;
- $T_{n \to n-1}$ rappresenta la trasformazione spaziale tra I_n e I_{n-1}.

Se sono conosciuti $D_n(p_n)$ e $T_{n \to n-1}$ allora è possibile ottenere le corrispondenze tra pixel nelle immagini I_n e I_{n-1}, come viene mostrato in figura 2.9(a).

Zhou et al. [49] propongono una rete, mostrata in figura 2.9(b), che stima la mappa di profondità $\hat{D}_n$ da una singola immagine I_n e una rete che ricava la trasformazione $\hat{T}_{n \to n-1}$ tra frame successivi. L'equazione 2.16 viene modificata come segue:



(a) (b)

Figura 2.9: (a) Ricostruzione del frame $\hat{I}_n(p)$ a partire da I_{n-1}. (b) Framework della rete neurale non supervisionata per la stima di profondità. Durante il processo di training, la profondità D_n e la pose $\hat{T}_{n\to n-1}$ predette rispettivamente dalle due reti sono usate per stabilire la proiezione tra I_n e I_{n-1} così da ricostruire $\hat{I}_n$. Le differenze tra l'immagine reale I_n e la ricostruzione $\hat{I}_n$ vengono calcolate per effettuare la supervisione. Immagini prese da [48].

$$p_{n-1} \sim K\hat{T}_{n\to n-1}\hat{D}_n(p_n)K^{-1}p_n \tag{2.17}$$

La *reconstruction loss* è formulata come segue:

$$\mathcal{L}_{vs} = \frac{1}{N}\sum_{p}^{N} |I_n(p) - \hat{I}_n(p)| \tag{2.18}$$

dove p sono le coordinate pixel e $\hat{I}_n(p)$ denota il frame ricostruito. Viene introdotta in $\mathcal{L}_{vs}$ la funzione di loss *structural similarity* (SSIM) per quantificare la differenza tra immagine ricostruita e target:

$$\mathcal{L}_{vs} = \alpha\frac{1 - SSIM(I_n - \hat{I}_n)}{2} + (1-\alpha)|I_n - \hat{I}_n| \tag{2.19}$$

dove α è il fattore di bilancio. Inoltre, Godard et al. [15] hanno dimostrato che, durante la fase di training, calcolare il valore minimo dell'errore di ricostruzione rispetto a calcolare la media permette alla rete di apprendere meglio. A partire dai risultati di Monodepth[14], vengono definite: una nuova funzione di *appearance matching loss* per affrontare il problema dei pixel occlusi, una nuova funzione di *auto-masking* per ignorare i pixel statici durante il movimento della camera e una nuova funzione di *multi-scale appearance matching loss* per ridurre gli artefatti nella stima di profondità.

Viene applicato l'algoritmo di ricostruzione della vista del frame $\hat{I}_n(p)$ a partire da I_{n-1}. Viene adottata una funzione di loss *edge-aware depth smoothness* simile a [14] per ottenere una mappa di profondità consistente a livello locale:

$$\mathcal{L}_{smooth} = \frac{1}{N}\sum_{p}^{N} |\nabla D(p)| \cdot (e^{-|\nabla I(p)|)^T} \tag{2.20}$$

dove T si riferisce all'operatore di trasposizione. Anche se, durante il processo di training, la rete che effettua la stima di profondità è associata alla rete che effettua il calcolo della pose, ovvero la posizione e l'orientamento dell'immagine rispetto ad un sistema di coordinate, durante l'inferenza possono essere utilizzate indipendentemente l'una dall'altra. Le due equazioni 2.17 e 2.20 formano il framework base dei metodi non supervisionati.

Metodi basati su maschera di oggetti dinamici

L'algoritmo di ricostruzione della vista, che si basa sulla funzione di proiezione, assume che la scena sia statica. Tuttavia, la posizione degli oggetti dinamici sui frame adiacenti non soddisfa le assunzioni precendenti andando ad impattare negativamente sulla stima dell'errore e di conseguenza sul training della rete. Vengono utilizzate delle maschere che riducono l'influenza degli oggetti dinamici e delle occlusioni sulla loss di ricostruzione $\mathcal{L}_{vs}$. Casser et al. [4] riducono gli effetti degli oggetti dinamici modificando la loss come segue:

$$\mathcal{L}_{vs}^{M} = \frac{1}{N}\sum_{p}^{N} M|I_n(p) - \hat{I}_n(p)| \tag{2.21}$$

dove M si riferisce alla maschera degli oggetti dinamici precedentemente predetta da una rete.

Metodi basati su visual odometry

Invece di utilizzare la pose stimata da una rete, viene usata per assistere la stima della profondità la pose ottenuta dalla stima di spostamento di un robot nel tempo utilizzando le sequenze immage, chiamata *Visual Odometry* (VO). Wang et al. [45] utilizzano direttamente la visual odometry in unione alla mappa di profondità generata dalla rete per stimare la pose tra frame vicini minimizzando l'errore fotometrico. Quindi, le pose calcolate in questo modo vengono riusate nella rete. In questo modo, viene migliorata l'accuratezza della stima di profondità avendo la rete supervisionata da pose più precise.

Metodi basati su framework multi-task

Approcci recenti introducono reti aggiuntive multi-task nel framework base per i task di *optical flow*, *object motion* e calcolare i parametri intrinseci della camera. Nel

lavoro di Casser et al. [4] viene proposta una rete in grado di modellare il moto degli oggetti, prendendo in input delle immagini segmentate. Tutti i metodi citati fino a questo punto si basano sull'assunzione che siano conosciuti i parametri intrinseci delle camere. Li et al. [25] utilizzano la stessa architettura di rete encoder-decoder di [49] per la rete che stima la profondità. Inoltre migliorano la rete che effettua il calcolo della pose di [17], dando la possibilità di stimare i parametri intrinseci della camera con lo scopo di ridurre i prerequisiti nella fase di training.

2.3.3 Metriche per la valutazione dell'errore

In letteratura le principali e più diffuse metriche per la valutazione dell'errore nella stima di profondità si basano sui lavori di Karsch, Liu e Kang [20], Ladicky, Shi e Pollefeys [21] e Eigen, Puhrsch e Fergus [10].
Karsch, Liu e Kang utilizzano le seguenti metriche per definire il grado di errore della predizione rispetto alla ground truth:

$$\textit{absolute relative error } \textbf{(abs rel)} = \frac{|D - D^*|}{D^*} \tag{2.22}$$

$$\log_{10} \textit{ error} = |\log_{10}(D) - \log_{10}(D^*)| \tag{2.23}$$

$$\textit{root mean squared error } (\textbf{RMSE}) = \sqrt{\frac{\sum_{i=1}^{N}(D_i - D_i^*)^2}{N}} \tag{2.24}$$

dove D e D^* sono rispettivamente la profondità stimata e ground truth. L'absolute relative error e $\log_{10}$ error valutano il grado di approssimazione tra il dato reale e quello predetto utilizzando due diverse scale. RMSE è la radice quadrata della media degli errori al quadrato. L'effetto di ciascun errore su RMSE è proporzionale alla dimensione dell'errore al quadrato; quindi errori più grandi hanno un effetto maggiore sull'RMSE, rendendola sensibile ai valori outlier. Varianti di queste metriche sono:

$$\textit{squared relative error } (\textbf{sq rel}) = \frac{(D - D^*)^2}{D^*} \tag{2.25}$$

$$\textit{root mean squared error } \log (\textbf{RMSE log}) = \sqrt{\frac{\sum_{i=1}^{N}(\log(D_i) - \log(D_i^*))^2}{N}} \tag{2.26}$$

Il lavoro di Ladicky, Shi e Pollefeys [21], invece, definisce una metrica di accuratezza. Viene definita l'equazione che descrive il *maximum relative error* δ come:

$$max(\frac{d_{gt}}{d_{res}}, \frac{d_{res}}{d_{gt}}) = \delta < \delta_{POS} \tag{2.27}$$

dove d_{res} è la profondità stimata, d_{gt} è la ground truth e δ_{POS} rappresenta una soglia *threshold* di accettazione dell'errore (di norma viene impostata a 1.25). Questa metrica ha come dominio l'intervallo $[0, 1]$ che, se espresso in percentuale, rappresenta la quantità di punti predetti che hanno distanza massima dalla ground truth minore della soglia. Varianti di questa metrica considerano il parametro *threshold* elevato al quadrato e al cubo: δ^2_{POS} e/o δ^3_{POS}.
Eigen, Puhrsch e Fergus[10] definiscono una metrica invariante rispetto all'errore di scala. Per una mappa di profondità predetta y e un valore di ground truth y^*, entrambe con n pixel, viene definita la metrica *scale-invariant mean squared error*, in scala logaritmica, come:

$$D(y, y^*) = \frac{1}{2n}\sum_{i=1}^{n}(\log y_i - \log y_i^* + \alpha(y, y^*))^2 \tag{2.28}$$

dove

$$\alpha(y, y^*) = \frac{1}{n}\sum_{i}(\log {y_i}^* - \log y_i) \tag{2.29}$$

è il valore che minimizza l'errore per un dato (y, y^*).

2.4 Rappresentazione dell'ambiente di navigazione

Dato che per valutare la stima di profondità degli approcci analizzati in questo lavoro di tesi sono stati sfruttati dati di sensori che misurano la geometria dell'ambiente, di seguito vengono mostrati diversi tipi di rappresentazione dello spazio di navigazione di un robot: mappe 2D, mappe 3D e mappe 2.5D.

2.4.1 Mappe 2D

Le mappe 2D, chiamate *griglie di occupazione*, rappresentano l'ambiente sotto forma di griglia bidimensionale. La loro rappresentazione è simile alle cartine stradali dove si ha una vista dall'alto (*bird's eye view*) dell'ambiente di navigazione. La griglia è costituita da celle, ognuna delle quali è una rappresentazione discreta di una porzione di spazio. Ad ogni cella viene associato un stato: libero, occupato o sconosciuto. Se consideriamo la griglia come una immagine in scala di grigi allora ogni cella è un pixel e il suo valore è 0 se è occupata, 255 se è libera o assume un valore intermedio

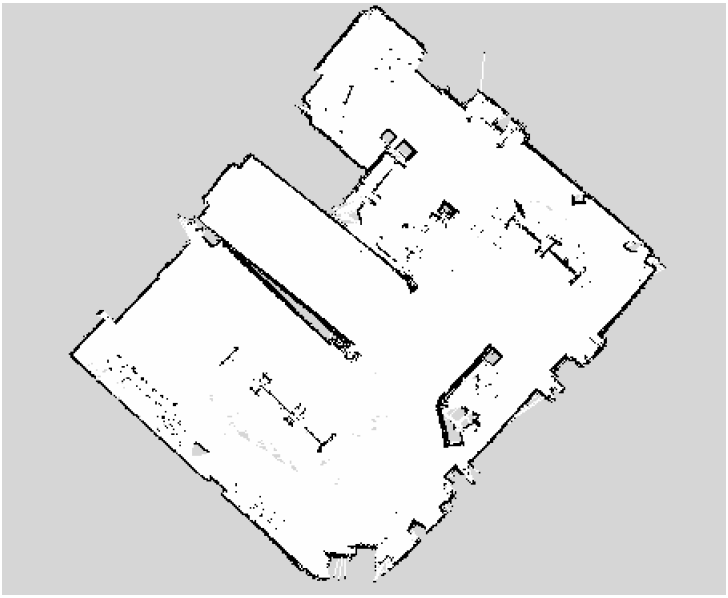


Figura 2.10: Griglia di occupazione di un ambiente al chiuso. I pixel bianchi rappresentano lo spazio libero, mentre quelli neri quello occupato, come muri o oggetti.

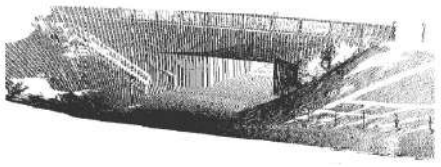
se non si ha informazione. In figura 2.10 viene mostrata un esempio di griglia di occupazione in un ambiente al chiuso.
Nel caso invece si considerino griglie probabilistiche la mappa è formata da celle contenenti valori di probabilità che rappresentano l'*occupancy probability*, ovvero la probabilità di occupazione di quella cella. Un valore vicino a 0 rappresenta uno spazio libero, un valore vicino a 1 uno spazio occupato, mentre un valore vicino a 0.5 rappresenta incertezza sullo stato della cella.
In generale, prima di creare una griglia di occupazione si cerca di rispettare queste tre assunzioni:

- le celle possono essere nello stato libera o occupata;
- le celle sono indipendenti le une con le altre;
- si assume che il mondo sia statico.

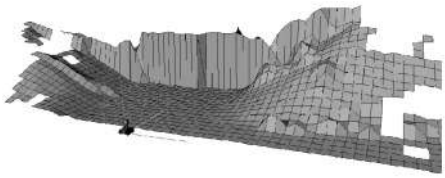
Utilizzando sensori come possono essere LIDAR, sonar o camere stereoscopiche è possibile trasformare l'informazione acquisita in una griglia di occupazione dell'ambiente. Ad oggi, la maggior parte dei robot mobili in ambienti al chiuso usa mappe di questa tipologia per navigare, localizzarsi e pianificare il percorso.

2.4.2 Mappe 2.5D

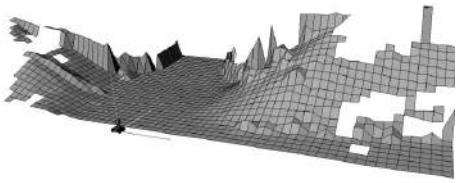
Il secondo modello di rappresentazione considerato sono le mappe 2.5D, chiamate *mappe di elevazione*. Le mappe di elevazione si basano sulla struttura a griglia di oc-



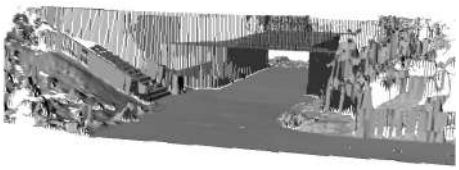
(a) Pointcloud.



(b) Mappa di elevazione.



(c) Mappa di elevazione estesa.



(d) Mappa di elevazione multi livello.

Figura 2.11: Rappresentazione di un ponte utilizzando diversi modelli. Immagine presa da [42].

cupazione vista in 2.4.1 ma, poiché vengono utilizzate principalmente in condizioni di superfici non piatte, all'intero di ogni cella è presente anche il valore di altitudine della superficie nel punto corrispondente dell'ambiente. In questa tipologia di ambiente le mappe di elevazione hanno il vantaggio di avere una rappresentazione compatta, avendo un peso trascurabile in memoria, e allo stesso tempo rendono possibile effettuare la navigazione e la localizzazione. Tuttavia, una limitazione di questo modello è dato dal fatto che ogni cella rappresenta una sola superficie e quindi non è possibile rappresentare strutture verticali non omogenee o diversi livelli di superficie (ad esempio oggetti appesi). Di conseguenza, in letteratura, per aumentarne l'espressività sono state proposte diverse estensioni. Pfaff, Triebel e Burgard hanno proposto un approccio che estende le mappe di elevazione con il riconoscimento di oggetti verticali e appesi, con anche un dislivello notevole nell'altezza [35]. Un successivo approccio [42] presenta una nuova rappresentazione, denominata *multi-level surface maps* (MLS maps). In ogni cella della griglia vengono salvati i valori di altitudine di superfici multiple, chiamati patches, che vengono poi utilizzati per rappresentare strutture verticali anche complesse. Un esempio delle diverse tecniche appena descritte è mostrato in figura 2.11.

2.4.3 Mappe 3D

Un altro modo per rappresentare lo spazio sono le mappe 3D che, a differenza delle griglie di occupazione, sono più espressive in quanto l'ambiente viene rappresentato tridimensionalmente. In letteratura sono presenti diverse rappresentazioni, alcune di queste sono: *pointcloud*, *voxel grid* e *octomap*.

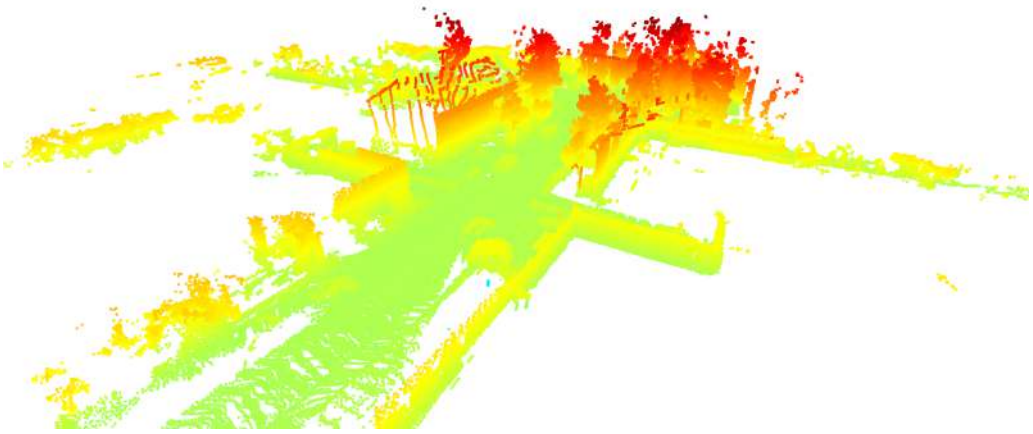


Figura 2.12: Pointcloud LIDAR di un ambiente urbano.

Pointcloud

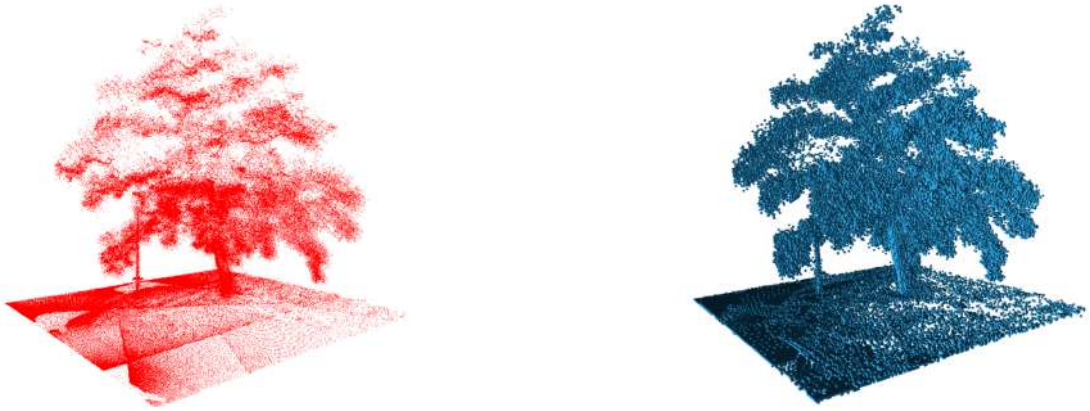
Il primo tipo di rappresentazione che ha il vantaggio di non effettuare discretizzazioni dei dati si chiama *pointcloud*. Una pointcloud, come si può vedere in figura 2.12, è un insieme di punti nello spazio che rappresenta l'ambiente in tre dimensioni. Ogni punto della pointcloud è definito rispetto ad un sistema di riferimento in coordinate cartesiane (X, Y, Z). Le mappe pointcloud possono essere generate tramite l'allineamento di scansioni acquisite ad istanti di tempo e posizioni diverse da sensori come LIDAR o scanner 3D. Il processo di allineamento di diverse pointcloud prende il nome di *pointcloud registration*.

Voxel Grid

Il secondo tipo di rappresentazione consiste nel discretizzare l'ambiente creando una griglia regolare di cubi di uguale volume, chiamata *voxel grid*. La voxel grid può essere considerata come la versione in tre dimensioni della griglia di occupazione. Di conseguenza ogni voxel rappresenta un singolo pezzo di informazione. Se non c'è informazione, il voxel non viene definito. Le differenze tra pointcloud e voxel sono mostrate in figura 2.13.

Octomap

Le due rappresentazioni sopra descritte, pointcloud e voxel grid non permettono di distinguere lo spazio libero o occupato da quello sconosciuto. Del resto, la libreria *Octomap* [19], presentata come evoluzione delle voxel grid da Hornung et al., permette di modellare esplicitamente lo spazio libero. L'approccio di questa rap-



(a) Pointcloud. (b) Voxel grid.

Figura 2.13: Differenze tra pointcloud e voxel grid. La discretizzazione dei voxel limita il grado di precisione nella rappresentazione dell'ambiente ma riduce la sparsità. Immagini prese da [19].

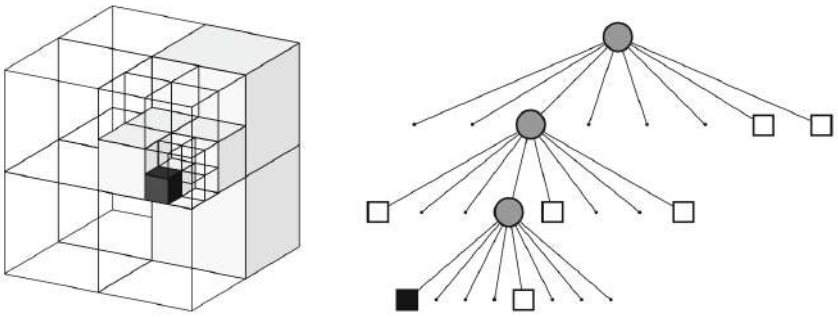


Figura 2.14: Esempio di octree, i nodi/cubi bianchi rappresentano lo spazio libero, quelli neri lo spazio occupato. (A sinistra) Il modello volumetrico. (A destra) La rappresentazione ad albero. Immagine presa da [19].

presentazione si basa su una struttura ad albero chiamata *octrees* [16] che viene utilizzata per stimare lo stato di occupazione dei voxel. Un octree è una struttura dati gerarchica per la suddivisione dello spazio in 3D. Ogni nodo in un octree rappresenta lo spazio contenuto in un volume cubico, il voxel. Questo volume è ricorsivamente suddiviso in otto sotto volumi fino a che viene raggiunta la dimensione minima del voxel, che determina la risoluzione minima dell'octree. Un esempio di octree viene mostrato in figura 2.14. Inoltre, grazie al fatto che l'octree è una struttura gerarchica, è possibile tagliare l'albero a diversi livelli per ottenere una suddivisione più o meno grossolana della mappa. Un esempio di octomap sezionata a diverse risoluzioni viene mostrata in figura 2.15.

Figura 2.15: Esempio di octomap. Limitando la profondità dell'albero si possono ottenere risoluzioni diverse della stessa mappa. (Da destra) Voxel di 1.28 m, poi di 0.64 m e infine di 0.08 m di lato. Immagine presa da [19].

2.5 Dataset

Nel campo della robotica mobile la ricerca è strettamente dipendente da vaste quantità di dati. Nel corso degli anni, la comunità scientifica ha messo a disposizione diversi dataset grazie ai quali è possibile sviluppare tecniche per affrontare i problemi inerenti a questo ambito. Inoltre, con la rapida crescita di tecniche basate su reti neurali, l'acquisizione di grandi quantità di esempi etichettati è risultata molto importante per poter allenare modelli in grado di effettuare delle predizioni accurate. Quindi, risulta molto importante saper selezionare il dataset corretto per il task che si vuole approcciare. In questa sezione viene effettuata una rassegna di dataset utilizzati comunemente in letteratura per diversi task di robotica mobile.

KITTI

Il dataset di KITTI [13] è tra i più grandi e comuni dataset utilizzati per task di computer vision come optical flow, visual odometry, riconoscimento di oggetti, segmentazione semantica e tracking di oggetti nella scena. Inoltre, è anche il benchmark più comune in letteratura per training e testing di reti neurali per la stima di profondità da singola immagine. Il dataset contiene diverse sequenze di immagini etichettate in base al luogo di acquisizione (ad esempio "city", "residential" e "road"). Ogni sequenza contiene i dati acquisiti dalle immagini stereoscopiche, la scansione LIDAR e il dato GPS. Poichè sono stati acquisiti in 11 sequenze i dati ground truth dell'odometria, questo dataset viene anche utilizzato per la valutazione di algoritmi di stima della pose.

Oxford Robotcar

Oxford Robotcar [30] è un dataset acquisito in ambiente urbano che consiste in immagini, scansioni LIDAR, dati GPS e dei sensori inerziali (IMU). Il dataset è composto da una moltitudine di sequenze di attraversamenti dello stesso percorso nell'arco del periodo di un anno per un totale di oltre 1000km di dati acquisiti. Di conseguenza, Oxford Robotcar viene molto utilizzato per task di *place recognition*, ovvero il task di riconoscere un luogo al variare del punto di vista o degli oggetti dinamici presenti, associando, ad esempio, il dato immagine con la mappa LIDAR.

Cityscapes

Il dataset Cityscapes [8] si concentra principalmente su task di segmentazione semantica. Il dataset contiene dati acquisiti da una coppia di camere in conformazione stereo. Per il task di segmentazione semantica gli autori del dataset hanno voluto creare un insieme di 5000 immagini annotate in modo preciso e un altro insieme più grande di 20000 immagini con una annotazione più grossolana. Poiché in questo dataset non è presente un sensore LIDAR, il valore di profondità ground truth viene ottenuto dai dati immagine stereo ricavando la mappa di disparità. In letteratura Cityscapes viene utilizzato per il pre-training di reti neurali supervisionate che saranno poi successivamente allenate su un diverso dataset. Il dataset di training racchiude al suo interno 22973 coppie di immagini acquisite in diverse città con una grande varietà di scene urbane.

Make3D

A differenza dei dataset descritti in precedenza, Make3D [38] contiene al suo interno solo immagini RGB-D acquisite tramite una camera monoculare. Poichè al suo interno non sono presenti sequenze video monoculari ma immagini sparse, e quindi, non si hanno frame ad istanti di tempo adiacenti, i metodi non supervisionati non lo utilizzano come dataset per la fase di training. Tuttavia, è possibile utilizzarlo come dataset di test per valutare l'abilità di generalizzazione di reti allenate su altri dataset.

NYU Depth

Il dataset NYU Depth [41] contiene al suo interno una grande varietà di scene acquisite in ambienti al chiuso. Utilizzando un altro approccio rispetto a dataset come KITTI e Oxford Robotcar, all'interno di NYU sono presenti sequenze di video monoculari, con le relative immagini di profondità, acquisite tramite un sensore camera

RGB-D. Tuttavia, a causa della diversa frequenza di acquisizione tra la camera RGB e la camera di profondità, non sempre è presente una corrispondenza tra la mappa di profondità e l'immagini RGB. Di conseguenza è necessario allineare la mappa con l'immagine. Per effettuare questo l'allineamento ciascuna mappa di profondità viene inizialmente associata all'immagine RGB con il timestamp più vicino. Successivamente, l'allineamento viene rifinito utilizzando le relazioni geometriche tra camera RGB e camera di profondità. Una volta effettuato l'allineamento non è detto che tutti i pixel abbiano un valore di profondità corrispondente, quindi, i pixel senza valore vengono mascherati durante gli esperimenti.

Capitolo 3

Descrizione del lavoro svolto

Conclusa la rassegna sulla letteratura e lo stato dell'arte, in questo capitolo vengono illustrate le motivazioni che hanno portato alla realizzazione del lavoro oggetto di tesi. Successivamente, viene fornita una descrizione ad alto livello del processo sviluppato per affrontare il problema della valutazione di stima di profondità. Viene illustrata nel dettaglio la nuova pipeline di valutazione e, infine, vengono descritti i dataset considerati con le relative modifiche effettuate.

3.1 Motivazioni

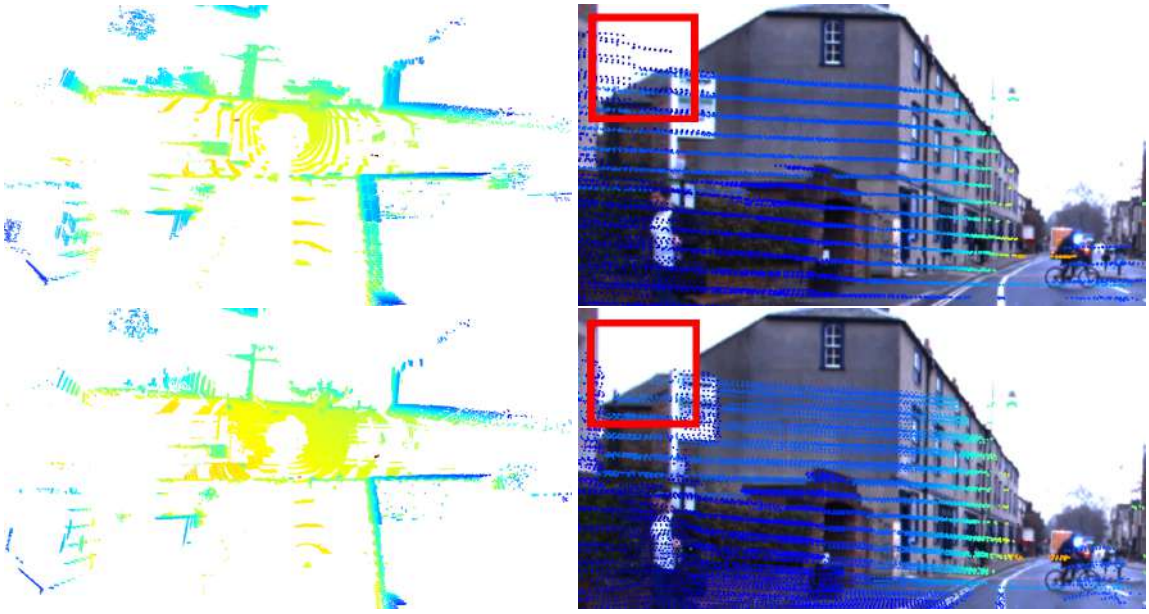
In questo lavoro di tesi, l'obiettivo principale è lo sviluppo di una metrica di valutazione per reti neurali che stimano la profondità di una scena partendo da una singola immagine. Le tecniche di valutazione esistenti in letteratura [10] si basano sul confronto tra il valore di profondità associato al punto della mappa LIDAR con il valore predetto del relativo pixel immagine. In letteratura, nei dataset maggiormente utilizzati, la singola scansione laser è piuttosto sparsa e, spesso, la posizione del sensore LIDAR sulla piattaforma non permette di scansionare parte della scena ottenendo la maggior parte dei punti racchiusa in un breve intervallo di distanze dall'osservatore. Per molti punti che vengono ricostruiti dalla rete non sono disponibili punti nella ground truth. Di conseguenza, quando si effettua la valutazione di queste reti utilizzando le tecniche esistenti in letteratura, i risultati che si ottengono forniscono una valutazione parziale in quanto parti considerevoli della scena non sono valutabili. Per superare questo limite in questo lavoro viene presentata una nuova metodologia di valutazione delle prestazioni che si basa sull'allineamento di più scansioni LIDAR per ottenere una ground truth più ricca in un intervallo di profondità maggiore. In questo modo, nella successiva valutazione della predizione è possibile confrontare il valore 3D ricostruito con i punti di una ground truth più

densa, avendo una valutazione migliore anche su oggetti molto distanti rispetto al punto di vista.

3.2 Modellazione del problema

Partendo dalle premesse precedenti sono state identificate una serie di problematiche da affrontare prima di poter effettivamente valutare le prestazioni delle reti neurali. La prima problematica analizzata riguarda l'allineamento delle scansioni LIDAR. In particolare è necessario capire come concatenare le scansioni LIDAR per ottenere un miglioramento nella ground truth. Si è pensato, inizialmente, di effettuare un semplice allineamento sfruttando l'informazione data dal sensore di posizione GPS o dalla visual odometry (VO). Il processo di visual odometry è in grado di determinare la posizione e l'orientamento di un robot analizzando le immagini delle camere stereo. Tuttavia, mappe generate con la VO ottengono un errore di allineamento che aumenta con il crescere della distanza tra le pose delle scansioni. D'altra parte il dato GPS non peggiora in base alla distanza ma è soggetto a errori di misurazione che rendono difficile allineare perfettamente le scansioni. Un altro problema riguarda l'errore di proiezione dei punti della mappa nell'immagine, infatti andando a proiettare la mappa sul piano immagine utilizzando solo il GPS o la VO, è probabile la presenza di forti disallineamenti tra la rappresentazione RGB e la proiezione dei punti LIDAR.

Quindi, si è ricercata in letteratura una tecnica in grado di migliorare l'allineamento e in grado di ottenere una proiezione della mappa consistente rispetto all'immagine RGB. Si è deciso di allineare le diverse pointcloud utilizzando il processo di *pointcloud registration*. Applicando algoritmi di *pointcloud registration* su una coppia di pointcloud, ovvero un insieme di punti 3D, è possibile ottenere la trasformazione che esprime i punti di una pointcloud rispetto all'altra. Questo processo permette di correggere l'allineamento iniziale fatto tramite GPS o VO, come si può vedere in figura 3.1. Tuttavia, possono presentarsi due problemi se viene utilizzata questa tecnica per allineare scansioni a diversi istanti di tempo e in contesti in cui nella scena sono presenti oggetti dinamici. Il primo è dato dalla presenza di punti occlusi proiettati sul piano immagine, dovuti all'allineamento di scansioni acquisite in posizioni e orientamenti (pose) diversi. Il secondo problema è rappresentato da "strisciate", ovvero sequenze di punti all'interno della mappa, causate dal movimento di oggetti nella scena (ad esempio pedoni, veicoli e biciclette) nell'intervallo di acquisizione delle scansioni. In questo lavoro si è deciso di concentrarsi nel gestire il problema relativo alle occlusioni in quanto è quello presente in più contesti. Infatti,



(a) (b)

Figura 3.1: Differenza tra tecniche di allineamento: (in alto) VO e (in basso) VO + registrazione . (a) Pointcloud viste dall'alto. (b) Differenze nella proiezione dei punti 3D nell'immagine. Si può osservare il sostanziale disallineamento dei punti con i relativi pixel. In particolare, nel riquadro rosso vengono mostrati alcuni tra i punti in posizione errata dati dall'allineamento con solo VO.

le occlusioni, a differenza delle "strisciate", si riscontrano in scene sia statiche che dinamiche. La soluzione adottata per rimuovere le occlusioni consiste nell'utilizzare un filtro in grado di differenziare rispetto al punto di vista dell'osservatore camera quali punti della mappa proiettata nell'immagine sono visibili e quali invece sono occlusi, rimuovendoli. La fase finale del lavoro si è concentrata nel confrontare le mappe LIDAR allineate tramite registrazione con la mappa di profondità stimata dalle reti. L'errore di misura viene calcolato utilizzando le metriche di valutazione dello stato dell'arte [10] [20] [21].

3.3 Valutazione basata su mappa LIDAR

In questa sezione viene mostrata la pipeline di generazione di una mappa LIDAR in grado di poter essere utilizzata per la valutazione della stima di profondità ottenuta da reti neurali.
La pipeline è divisa in tre fasi:

1. allineamento di pointcloud;
2. proiezione dei punti 3D nel 2D;
3. filtraggio delle occlusioni.

3.3.1 Allineamento

Rispetto alle tecniche di valutazione esistenti basate su singola scansione LIDAR [10], l'approccio proposto utilizza una successione di scansioni prese in un intervallo di instanti di tempo successivi all'acquisizione dell'immagine. Concatenando più scansioni è possibile ottenere una proiezione più densa dei punti nell'immagine. La proiezione dei punti viene fatta in un piano immagine virtuale corrispondente al punto di vista della camera RGB. La prima tecnica utilizzata si basa sull'allineamento di pointcloud sfruttando solo l'informazione data dal sensore GPS o dalla visual odometry (VO). In ambienti urbani il dato GPS è poco affidabile in quanto il segnale degrada a causa della presenza di edifici che possono bloccarlo o sui quali rimbalza, fornendo delle rilevazioni rumorose della posizione. Di conseguenza, pointcloud allineate solo con il GPS mostrano visibili inconsistenze (ad esempio disallineamento degli edifici). Se si considera invece il sensore GPS RTK (Real-time kinematics) si possono ottenere allineamenti più precisi in quanto l'errore di posizionamento del veicolo viene corretto da una stazione base. Tuttavia, non viene risolto il problema legato agli edifici che in ambienti urbani degradano il segnale. Inoltre, a causa di

un costo superiore rispetto al normale GPS, i dataset presenti in letteratura non sempre contengono questo sensore. L'altro metodo per poter allineare pointcloud si basa sulla visual odometry. Quindi, utilizzando questo dato, è possibile allineare le scansioni. Tuttavia, anche in questo caso, non si ha una precisione nell'associazione dei punti tale da poter essere usata come ground truth. Infatti, l'errore di posizione e orientamento aumenta all'aumentare dei frame temporali considerati per l'allineamento, accumulandosi ad ogni successiva scansione considerata. Di conseguenza, per ridurre l'errore legato agli allineamenti con VO e alla degradazione del segnale GPS si è cercata una tecnica che portasse ad un allineamento più preciso delle scansioni. In letteratura, il processo di *pointcloud registration* è in grado di poter allineare pointcloud con una certa precisione. Per ogni coppia di scansioni, denominate *target* e *sorgente*, si calcola la trasformazione rigida, composta da rotazione e traslazione, che permette di allineare sorgente a target. Il primo algoritmo di registrazione che è stato utilizzato è *Iterative Closest Point* (ICP), algoritmo di ottimizzazione locale, presentato indipendentemente in tre diversi lavori [2][6][47]. Il relativo pseudo-codice è mostrato in 1. Avendo come obiettivo di ottenere la corretta trasformazione rigida che meglio allinea due pointcloud, l'esecuzione di ICP avviene iterativamente ripetendo le fasi fino a che si o raggiunge il massimo numero di iterazioni o le condizioni volute (ad esempio un errore sotto una certa soglia). Le fasi dell'algoritmo sono le seguenti:

1. effettuare il matching tra i punti della pointcloud sorgente con quella target;
2. trovare i valori per la trasformazione T che minimizza la distanza 3.1;
3. applicare la trasformazione T per allineare la scansione *sorgente* a quella *target*.

$$\sum_i w_i \|T \cdot y_i - m_i\|^2 \tag{3.1}$$

Per poter effettuare la registrazione è necessario rispettare due vincoli. Il primo richiede che le due scansioni siano in parte sovrapposte. Per superare questo vincolo viene introdotto un valore di *threshold* in grado di mediare tra convergenza e accuratezza. Un valore troppo piccolo potrebbe non far terminare l'algoritmo prima della fine delle iterazioni con la possibilità di ottenere una soluzione non accettabile, mentre un valore troppo grande potrebbero portare ad un allineamento poco preciso. Il secondo vincolo richiede invece che le pointcloud siano già parzialmente allineate.

In questo lavoro è stato utilizzato l'algoritmo nella sua variante di registrazione

Algorithm 1 ICP

Require: Due point cloud $X \in \mathbb{R}^{n\times 3}$, $Y \in \mathbb{R}^{m\times 3}$ e una trasformazione iniziale T_0

Ensure: La trasformazione T che allinea X e Y

$T \leftarrow T_0$
while *non converge* **do**
  for $i = 1$ **to** N **do**
    $m_i \leftarrow$ `FindClosestPointInX`$(T \cdot y_i)$
    if $\|m_i - T \cdot y_i\| \leq d_{max}$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end if
  end for
  $T \leftarrow \arg\min_T \{\sum_i w_i \|T \cdot y_i - m_i\|^2\}$
end while

punto-punto, ovvero ogni punto in una pointcloud viene confrontato con il più vicino punto nell'altra pointcloud, portando ad ottenere un buon allineamento delle scansioni ma a discapito di un costo elevato di tempo e computazione nel calcolo delle trasformazioni. Infatti, considerando che la dimensione media delle pointcloud è nell'ordine di 10^5 punti, sono necessarie diverse iterazioni prima di poter allineare con discreta precisione le scansioni. Per migliorare ancora di più la precisione dell'allineamento si è deciso di utilizzare Generalized ICP (G-ICP), variante di ICP sviluppata da Segal, Haehnel e Thrun [40]. A partire dall'algoritmo 1 sono state introdotte delle modifiche per ridurre la complessità e migliorare i risultati tramite l'aggiunta di un modello probabilistico. L'esecuzione di questo algoritmo sulle sequenze di coppie di pointcloud ha permesso di ottenere un risultato molto più accurato (ad esempio allineando in maniera molto precisa i singoli punti dei muri), ma a discapito di un aumento dei tempi di computazione. Tuttavia, dato che questo lavoro di tesi si propone di valutare degli approcci, non sono necessari tempi di calcolo real time in quanto il task di allineamento e registrazione avviene offline.

3.3.2 Proiezione dei punti 3D nel 2D

Per effettuare il confronto tra la ground truth e la predizione è necessario proiettare i punti 3D della mappa LIDAR in un piano immagine virtuale corrispondente al punto di vista dell'osservatore da cui si stima la profondità della scena. Quindi, considerando ora di aver ottenuto l'allineamento di una serie di scansioni LIDAR,

il passo successivo consiste nel proiettare i punti della pointcloud 3D nell'immagine 2D ottenendo per ogni punto le relative coordinate e valore di profondità. Prima di tutto, per effettuare una corretta proiezione, è necessario trasformare i punti 3D dal sistema di riferimento in cui sono espressi (LIDAR, mondo o del sensore di posizione) nel sistema di riferimento camera. Considerando, ora, di utilizzare il modello pinhole per rappresentare la camera, l'equazione che descrive la proiezione dei punti è data da:

$$p = MP \tag{3.2}$$

dove p sono le coordinate 2D pixel (x, y), M è la matrice di proiezione e P sono le coordinate del punto 3D (X, Y, Z). La matrice di proiezione M_{3x4} è data dal prodotto di due matrici:

$$M = KE \tag{3.3}$$

dove:

$$K = \begin{bmatrix} \frac{f}{h_x} & 0 & c_x \\ 0 & \frac{f}{h_y} & c_y \\ 0 & 0 & 1 \end{bmatrix}_{3x3}, \quad E = \begin{bmatrix} R_1 & -R_1T \\ R_2 & -R_2T \\ R_3 & -R_3T \end{bmatrix}_{3x4} \tag{3.4}$$

K_{3x3} è la matrice dei parametri intrinseci della camera dove $\frac{f}{h_x}$ e $\frac{f}{h_y}$ rappresentano la distanza del piano immagine dal centro del sistema di riferimento camera, c_x e c_y rappresentano le coordinate del centro di proiezione. E_{3x4} è la matrice dei parametri estrinseci della camera che rappresenta la relazione tra la camera e il sistema di coordinate mondo tramite le matrici di rotazione R_{3x3} e il vettore di traslazione T_{3x1}. Infine, per poter effettuare il prodotto descritto con la matrice di proiezione M_{3x4}, è necessario rappresentare il vettore punto P in coordinate omogenee $P_{4x1} = [P_{3x1}1]^T$. Come si è visto, l'equazione 3.2 definisce la trasformazione di un punto dallo spazio 3D al piano 2D. Per far sì che non si perda l'informazione di profondità ad ogni punto 2D viene associato il relativo valore di profondità. Vengono scartati tutti i punti che non ricadono nei limiti dati dalle dimensioni, altezza e larghezza, dell'immagine. Inoltre, se in un pixel vengono proiettati due o più punti, viene tenuto il punto con valore di profondità inferiore. Ciò che si ottiene al termine di questa fase è un matrice le cui dimensioni sono pari a quelle dell'immagine e in ogni cella è contenuto il valore di profondità.

3.3.3 Filtraggio delle occlusioni

Effettuando la registrazione di numerose pointcloud, la proiezione dei punti sul piano immagine può produrre una mappa di profondità con un alto grado di rumorosità. Infatti, è possibile che vengano proiettati punti occlusi rispetto al punto di vista

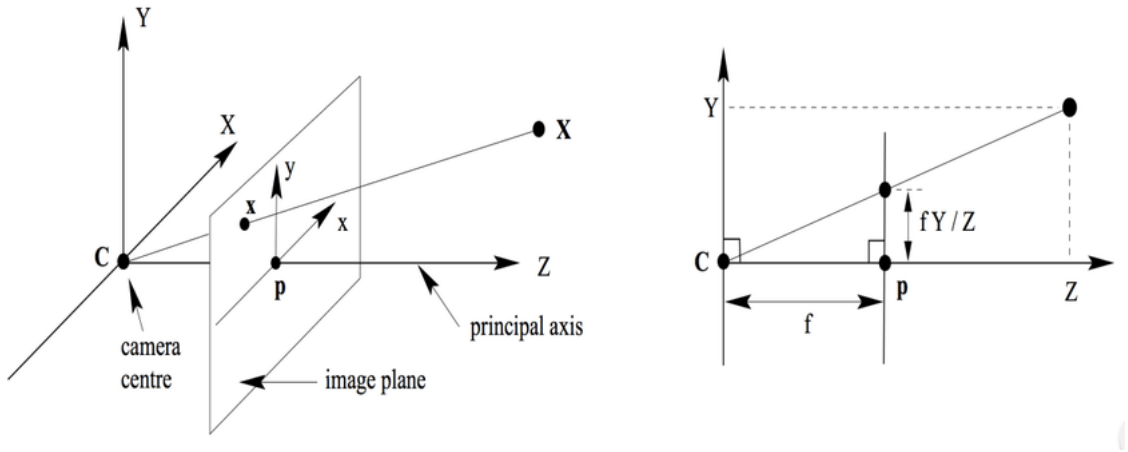


Figura 3.2: Proiezione dei punti 3D nel 2D.

Figura 3.3: Proiezione nell'immagine RGB di punti occlusi, posizionati dietro l'edificio, dovuta a diverse pose ed istanti di acquisizione delle scansioni. Immagine presa da [5].

dell'osservatore camera (ad esempio punti dietro edifici od altri ostacoli) a causa della diversa pose e diverso istante di acquisizione delle varie scansioni. Come si può vedere in figura 3.3, il movimento della piattaforma e l'acquisizione di scansioni a istanti di tempo successivo hanno causato la proiezione di punti che rispetto al punto di vista camera sono posti dietro l'edificio.

Per risolvere questo problema viene adottato il filtro di stima delle occlusioni presentato da Pintus, Gobbetti e Agus [36]. In figura 3.4 viene mostrato il funzionamento del filtro. Per ogni punto P_i viene costruito un cono sulla linea di proiezione che porta al centro immagine in modo tale che non intersechi nessun altro punto. Se il cono ha un diametro maggiore di un valore soglia (*threshold*) allora il punto P_i viene definito visibile, altrimenti viene considerato occluso e di conseguenza il suo valore di profondità viene posto a 0. In questo lavoro viene utilizzata una versione del filtro scritta in CUDA. Un esempio che riassume le fasi della pipeline viene mostrato in figura 3.5.

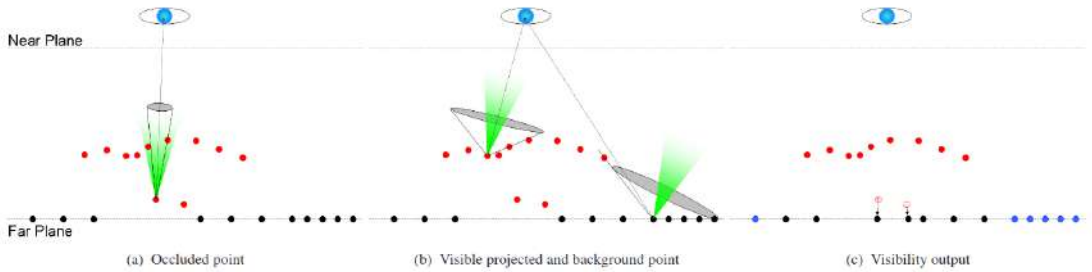
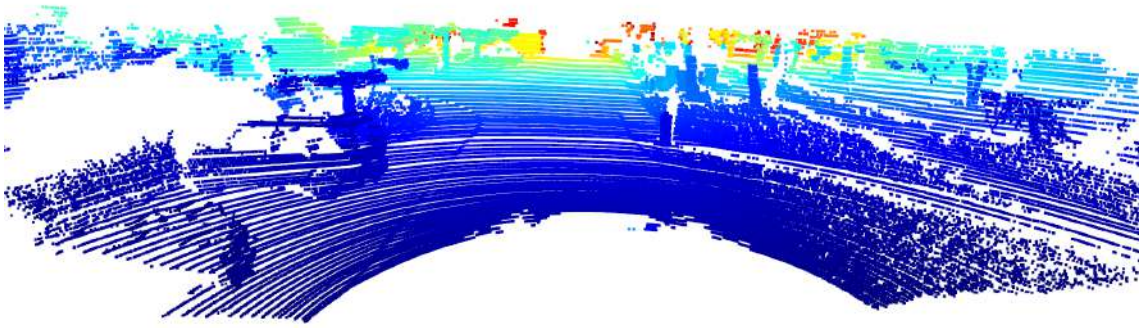


Figura 3.4: Funzionamento del filtro di occlusioni. I punti rossi e blu sono visibili, quelli neri sono occlusi. Il cono verde rappresenta l'angolo minimo per rendere un punto visibile. Immagine presa da [36].

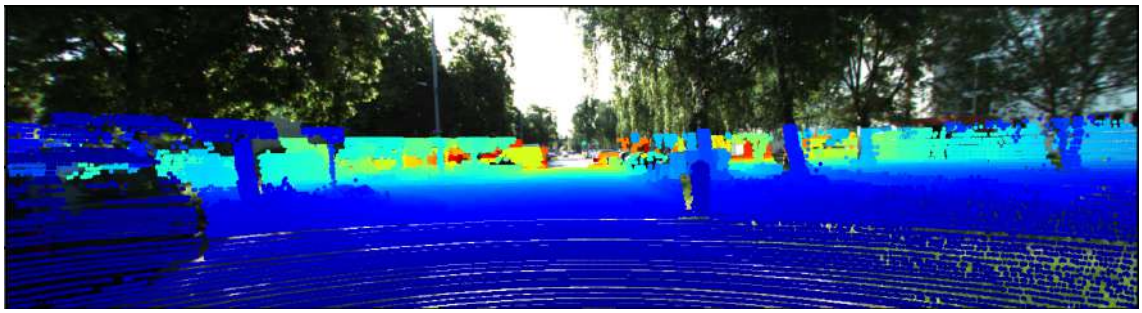
3.4 Dataset per la stima di profondità

La ricerca nel campo dei veicoli a guida autonoma è strettamente dipendente da vaste quantità di dati acquisiti nel mondo reale per lo sviluppo, l'analisi e la validazione di algoritmi prima di essere rilasciati e utilizzati su strade pubbliche. Nel corso degli anni sono stati rilasciati diversi dataset il cui focus primario consiste nel sviluppare competenze algoritmiche per la guida autonoma. Di conseguenza, una fase molto importante del lavoro è stata la ricerca di dataset che potessero andare bene per lo scopo della tesi. In particolare, si è mostrato necessario che tali dataset contenessero al loro interno le acquisizioni di camere (stereo o monoculari), le scansioni LIDAR (possibilmente utilizzando un laser multipiano) e un sensore che potesse essere in grado di rilevare la pose della piattaforma. Tali informazioni sono richieste in quanto è necessario allenare le reti su successioni di immagini e valutare le predizioni utilizzando la mappa ottenuta dall'allineamento di scansioni LIDAR. La scelta è quindi ricaduta su due dataset: KITTI [13] e Oxford Robotcar [30]. Infatti entrambi i dataset hanno le caratteristiche necessarie per poter essere utilizzati, avendo sequenze di immagini LIDAR e GPS. D'altra parte anche Cityscapes [8] è molto utilizzato in letteratura per valutare la stima di profondità, ma visto che non contiene i dato di profondità acquisito da un sensore laser non è possibile irrobustire la ground truth e di conseguenza non è stato considerato per questo lavoro.
In questa sezione viene effettuata una descrizione esaustiva dei dataset e per ognuno di essi vengono definite operazioni che hanno permesso la creazione della mappa ground truth formata dall'allineamento di più scansioni LIDAR.

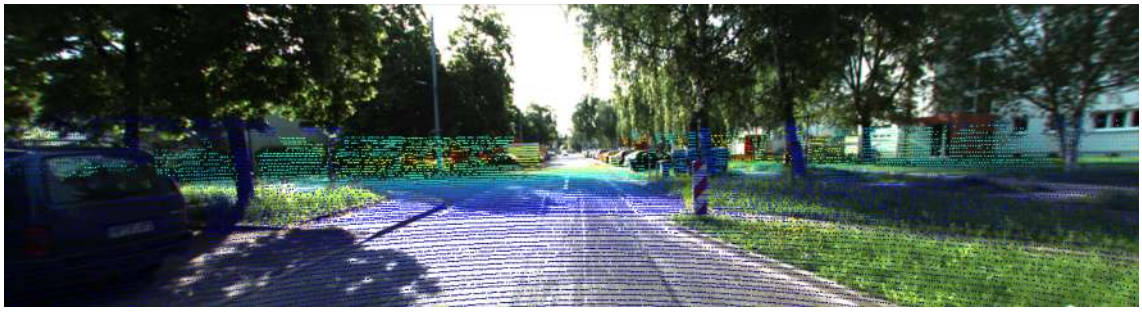
(a) Immagine RGB.



(b) Allineamento di pointcloud.

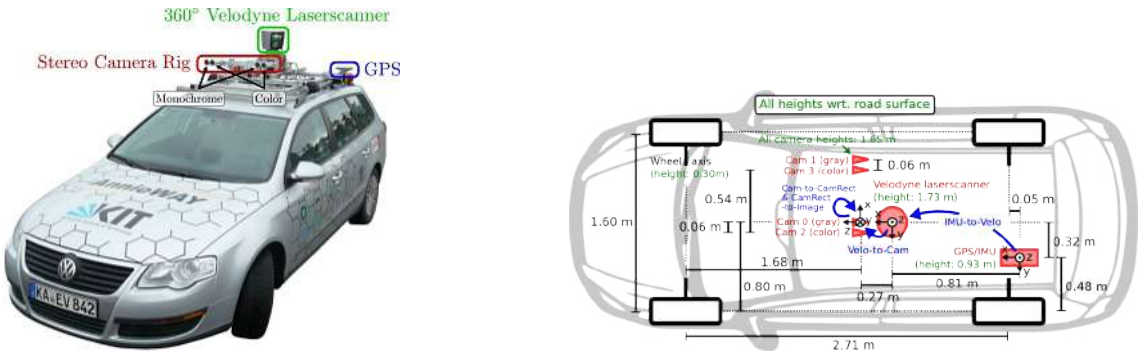


(c) Proiezione punti 3D nell'immagine 2D.



(d) Applicazione del filtro di occlusioni.

Figura 3.5: Fasi della pipeline di valutazione.



(a) Sensoristica. (b) Sistemi di riferimento.

Figura 3.6: Piattaforma KITTI.

3.4.1 KITTI

KITTI[1][13] è un dataset noto in letteratura e viene utilizzato come benchmark per la stima di profondità e di moto della camera all'interno di un ambiente (*ego-motion*). I dati sono stati raccolti utilizzando una piattaforma mobile, mostrata in figura 3.6, mentre veniva guidata nella zona di Karlsruhe in Germania. Le registrazioni dei dati sono avvenute durante le ore diurne durante le quali sono state acquisite scene di traffico reale, una varietà di contesti a partire da scenari autostradali per passare poi a luoghi rurali e infine a centri città, con una prevalenza di oggetti statici rispetto a quelli dinamici. La piattaforma è stata equipaggiata con camere a colori, a scala di grigi, GPS con sistema inerziale (IMU) e un LIDAR Velodyne a 64 piani di scansione. Per evitare la deriva nel tempo tutti i sensori sono stati calibrati e sincronizzati al termine di ogni giorno di registrazione. Viene utilizzato come riferimento il timestamp del Velodyne 3D, considerando ogni sua completa rotazione un frame temporale. Quando il laser è in posizione frontale viene attivata l'acquisizione della scena da parte delle camere. In questo modo viene minimizzata la differenza di osservazione tra il frame camera e la scansione laser causata dal moto della piattaforma, riducendo anche il disallineamento causato da ostacoli in movimento. Questa tecnica di sincronizzazione non può, però, essere applicata al GPS/IMU che di conseguenza viene sincronizzato scegliendo per ogni timestamp velodyne il timestamp GPS/IMU più vicino temporalmente. Con una frequenza di aggiornamento di 100 Hz da parte del GPS/IMU, questo metodo risulta in un errore medio tra i due sistemi di circa 5 ms. In unione al dataset viene fornito un kit di sviluppo[2] in linguaggio MATLAB e C++ per accedere e manipolare facilmente i dati. Un ulteriore dataset sviluppato dagli stessi autori di KITTI è KITTI-360[3][26]. Rispetto a

[1]http://www.cvlibs.net/datasets/kitti/
[2]http://www.cvlibs.net/datasets/kitti/raw_data.php
[3]http://www.cvlibs.net/datasets/kitti-360/index.php

KITTI, KITTI-360 contiene al suo interno sequenze con un dato GPS più accurato, mantenendo la stessa configurazione dei sensori sulla piattaforma mostrata in figura 3.6. Per il lavoro di tesi viene utilizzato questo dataset solo per la valutazione di metodi allenati con KITTI in quanto le immagini sono acquisite dagli stessi sensori camera ma KITTI-360 ha il vantaggio di avere una ground truth GPS più accurata, potendo allineare con più precisione le scansioni LIDAR. Per entrambi i dataset citati, la creazione della mappa LIDAR avviene allo stesso modo, in quanto i sensori, utilizzati per questo lavoro, sono posizionati nella stessa configurazione in entrambi i dataset. Il paragrafo successivo descrive in modo completo le operazioni necessarie per creare la mappa LIDAR per questi dataset.

Creazione della mappa LIDAR

L'idea alla base del processo di allineamento delle scansioni consiste nell'associare il frame immagine, che si vuole utilizzare per la predizione, con la scansione LIDAR acquisita all'istante di tempo più vicino. Viene, poi, effettuata la registrazione delle successive scansioni utilizzando la pose del GPS per allineare inizialmente le pointcloud. Prima di effettuare qualsiasi operazione di registrazione in ogni singola scansione vengono eliminati i punti in un intorno di 2.7 metri dal centro del sistema di riferimento. In questo modo vengono scartati tutti i punti acquisiti dal LIDAR che appartengono alla piattaforma, così da evitare di ottenere delle "strisciate" proiettate nell'immagine, causate dal movimento nel tempo. In figura 3.7 viene mostrata la pointcloud che si otterrebbe se non venissero eliminati questi punti.

Successivamente le singole scansioni vengono portate nel sistema di riferimento mondo (WORLD), come si vede in figura 3.8 (a), applicando prima la trasformazione che le porta da VELO al sistema GPS/IMU della piattaforma e poi la successiva da GPS/IMU a WORLD. In questo modo si ottengono pointcloud tutte nello stesso sistema di riferimento. Successivamente, è possibile effettuare la registrazione. Considerando le prime due scansioni P_0 target e P_1 sorgente, la registrazione con G-ICP calcola la trasformazione dei punti da sorgente a target. Applicando la trasformazione si ottiene P_1 allineata a P_0. Al passo successivo, target diventa P_1 e sorgente P_2. L'algoritmo continua iterativamente allineando ad ogni passaggio target e sorgente. Raggiunto un valore di soglia temporale (ad esempio dopo k=5 scansioni da quella iniziale) l'algoritmo termina ritornando la mappa LIDAR generata dall'allineamento e registrazione di tutte le scansioni con posizione e orientamento definiti nel sistema di riferimento mondo. Prima di effettuare la proiezione dei punti nell'immagine è necessario effettuare la catena di trasformazioni mostrata in figura 3.8 (b) per poter rappresentare i punti nel sistema camera usato per la proiezione.

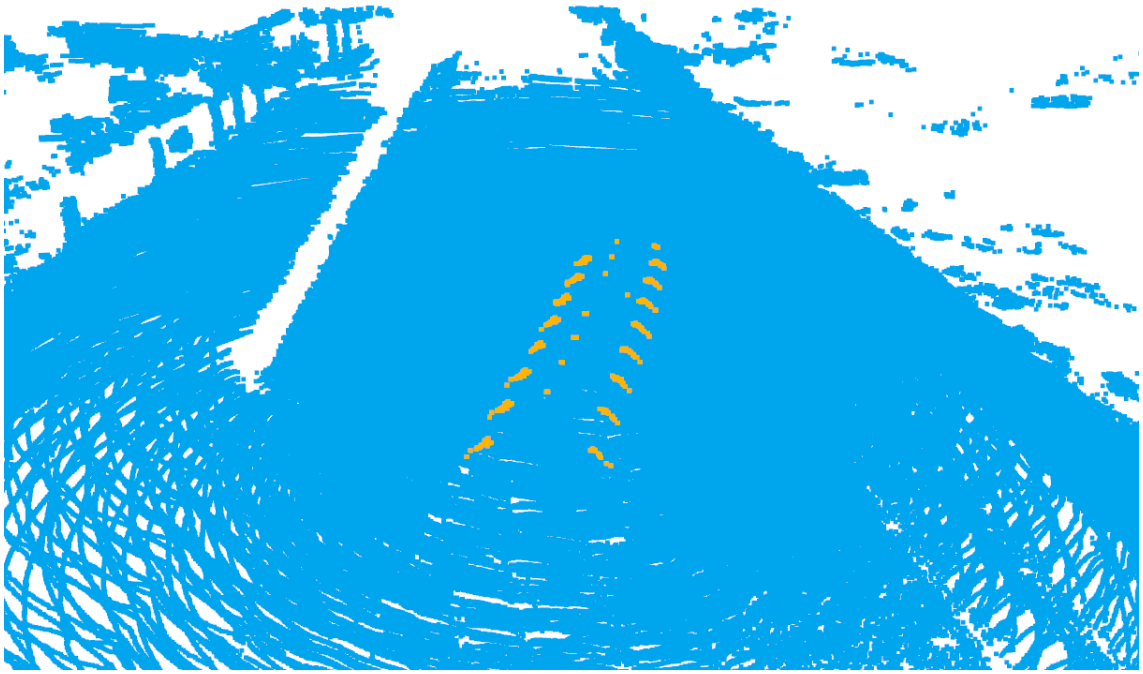


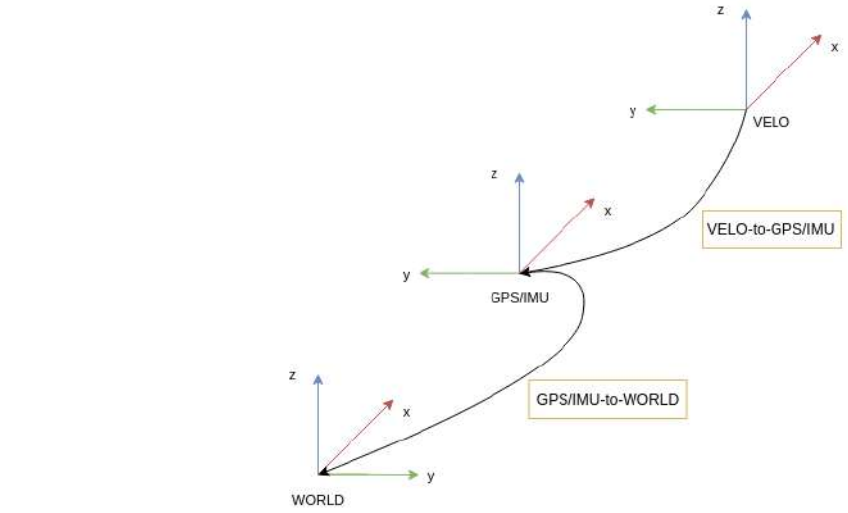
Figura 3.7: Esempio di allineamenti multipli, di scansioni del dataset KITTI, nelle quali non vengono eliminati i punti della piattaforma (in arancione).

Successivamente, avviene la proiezione dei punti 3D nell'immagine acquisita allo stesso istante temporale della prima scansione. Infine, viene applicato il filtro di occlusioni. Viene impostato il parametro soglia, *threshold*, che rappresenta il massimo angolo del cono proiettato, in modo tale da poter escludere la maggior parte dei punti occlusi. Il risultato della pipeline per alcuni esempi di KITTI viene mostrato in figura 3.9. Come si può vedere, il filtro riesce a discriminare bene i punti visibili da quelli occlusi.

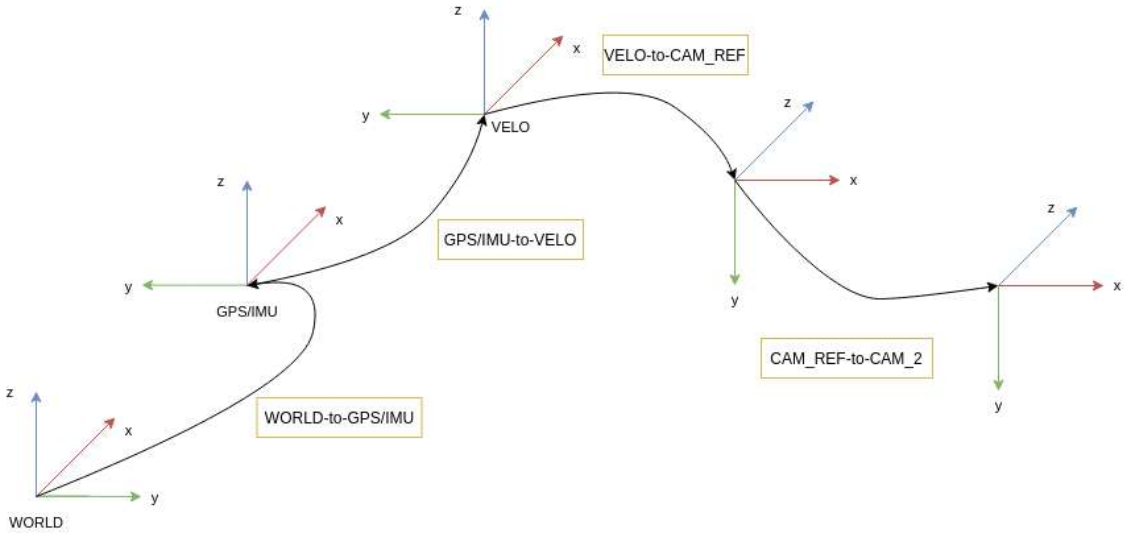
3.4.2 Oxford

Oxford RobotCar[4][30] è un dataset di grandi dimensioni che raggruppa una successione di attraversamenti multipli di una parte della città di Oxford. I dati di oltre 1000 km di guida sono stati raccolti utilizzando la piattaforma Oxford RobotCar, una automobile Nissan LEAF, con capacità di guida autonoma, mostrata in figura 3.10. La possibilità di poter effettuare più attraversamenti dello stesso percorso nell'arco del periodo di un anno con differenti condizioni ha permesso di acquisire un grande assortimento di variazioni nell'aspetto e nella struttura della scena dovuto a: cambiamenti nell'illuminazione, tempo atmosferico, oggetti dinamici, cambiamenti stagionali e lavori di edilizia. La piattaforma è stata equipaggiata con camere stereo

[4]https://robotcar-dataset.robots.ox.ac.uk/

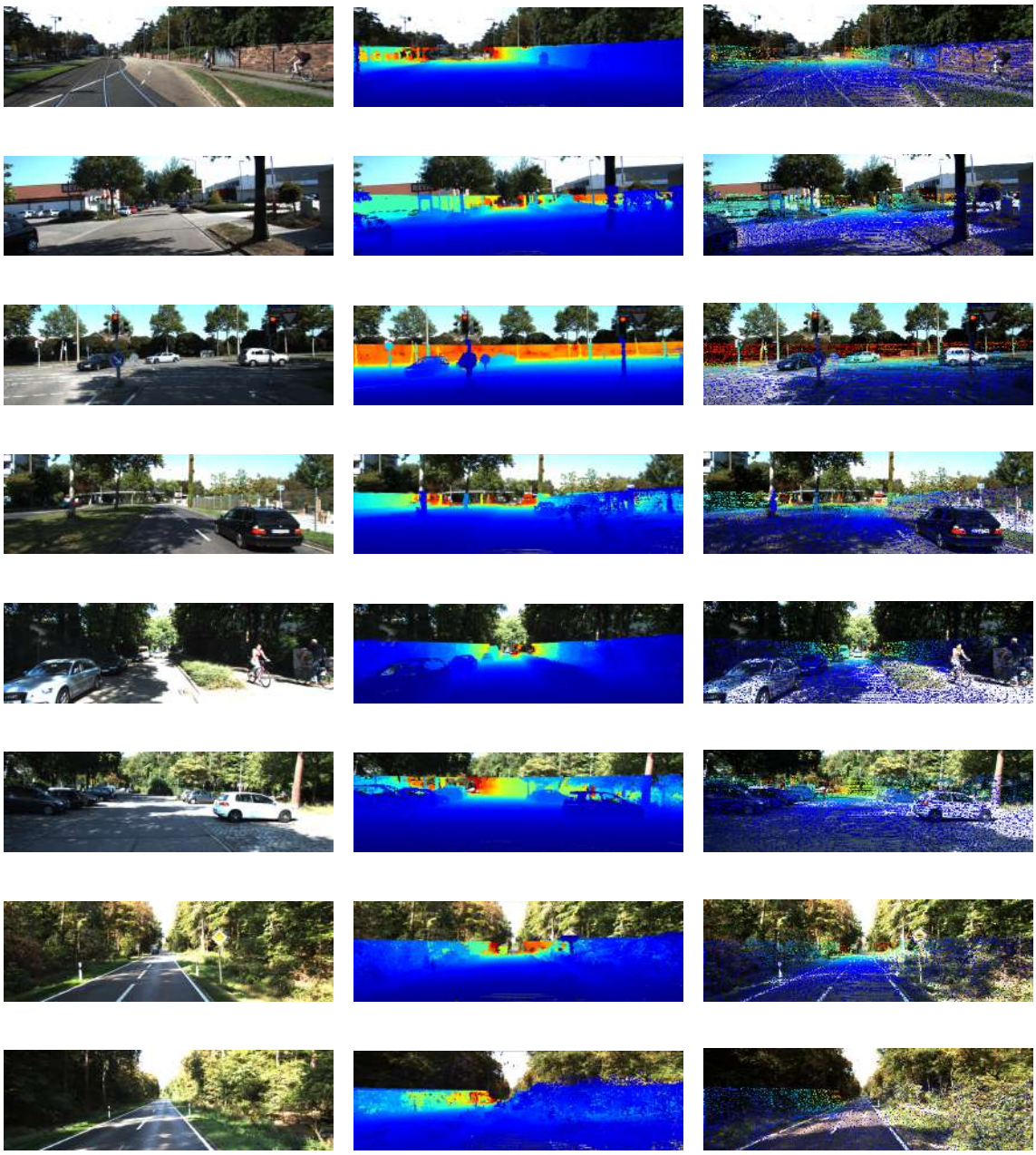


(a)



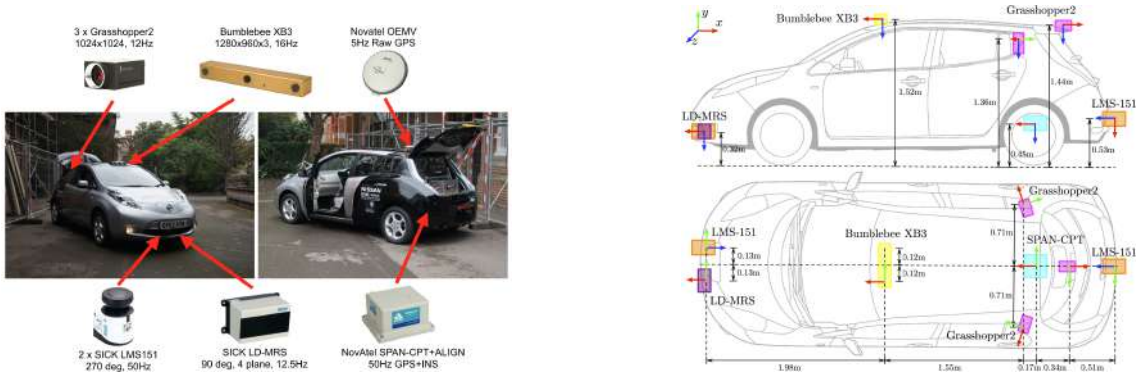
(b)

Figura 3.8: Catena di trasformazioni per il dataset KITTI e KITTI-360. (a) Trasformazione dei punti LIDAR dal sistema di riferimento VELO a WORLD per poter effettuare l'allineamento e la registrazione. (b) Trasformazione dei punti LIDAR dal sistema di riferimento WORLD a CAM_2 per poter effettuare una corretta proiezione nel 2D.



(a) (b) (c)

Figura 3.9: Esempi di immagini di KITTI processate utilizzando la nuova pipeline di valutazione. (a) Immagine RGB. (b) Proiezione dei punti 3D della mappa LIDAR nell'immagine. (c) Applicazione del filtro di occlusioni.



(a) Sensoristica. (b) Sistemi di riferimento.

Figura 3.10: Piattaforma Oxford RobotCar. Immagini prese da [30].

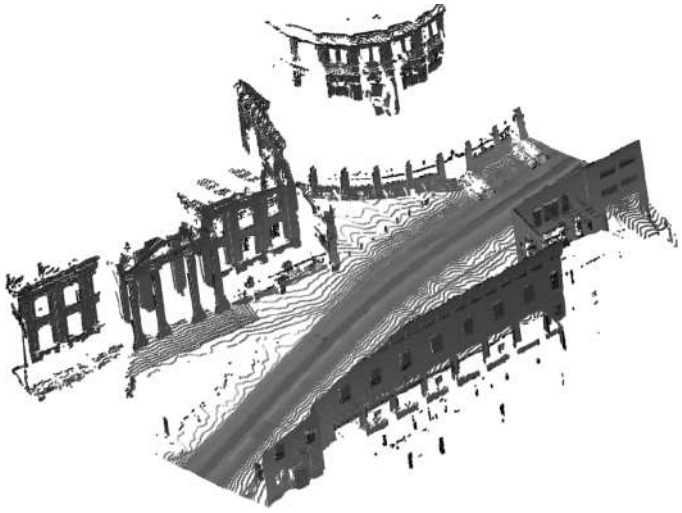


Figura 3.11: Pointcloud generatata a partire dalle scansioni LIDAR. Immagine presa da [30]. Immagine presa da [30].

in configurazione trinoculare, camere monoculari, sensori LIDAR sia a singolo che a quattro piani e un GPS con sistema inerziale (IMU). I sensori LIDAR a singolo piano sono montati nella parte anteriore e posteriore della piattaforma nella configurazione *"push-broom"* che permette di effettuare ad ogni istante di tempo una scansione verticale 2D dell'ambiente attorno al veicolo durante la marcia. Combinando queste registrazioni con una pose globale o locale è possibile ricostruire l'ambiente scansionato come indicato in figura 3.11. La qualità di ricezione delle antenne GPS e l'accuratezza del sistema inerziale varia in modo significativo durante l'arco di tempo nel quale sono stati acquisiti i dati portando a non essere un dato utilizzabile come ground truth per la validazione di algoritmi di localizzazione e *mapping*. Di conseguenza, le immagini stereo del sistema trinoculare vengono processate usando un sistema di visual odometry (VO) che fornisce le stime delle pose relative in riferimento alla pose sorgente locale. La soluzione ottenuta da questo

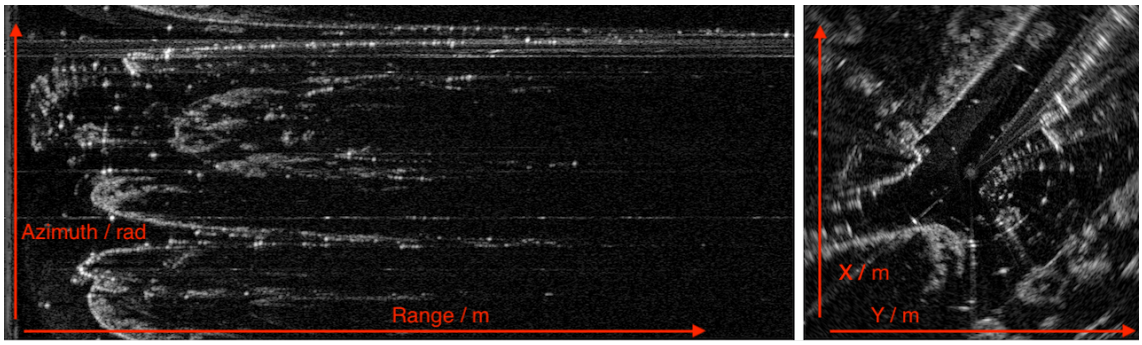
(a) GPS + INS. (b) Visual odometry.

Figura 3.12: Confrontro della stessa poincloud generata con il dato GPS+INS e VO. Immagini prese da [30].

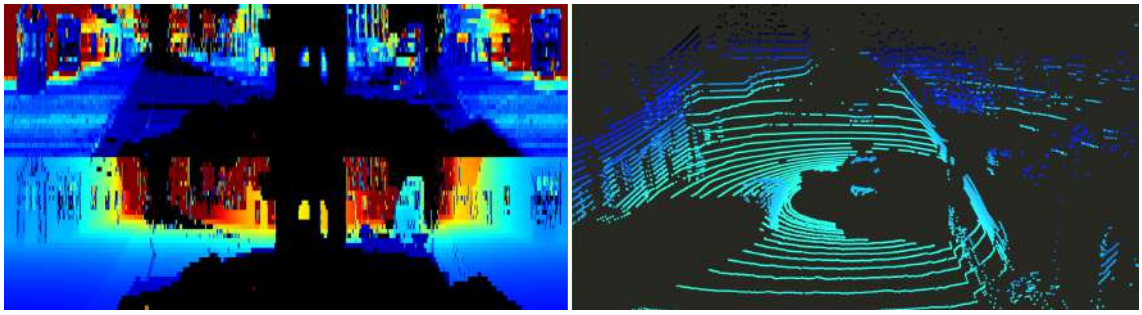
processo è abbastanza accurata per generare pointcloud 3D locali. Un confronto tra i due approcci appena descritti viene mostrato in figura 3.12.

In unione al dataset viene fornita una libreria[5] MATLAB e Python per accedere e manipolare facilmente i dati. Non essendoci una metodologia comune in letteratura su come approcciare il problema della valutazione della stima di profondità con questo dataset, si è deciso di provare diversi approcci. Per validare le predizioni è stato utilizzato il LIDAR a singolo piano. Tuttavia, poiché questo sensore genera una pointcloud 2D contenente solo punti vicini alla piattaforma, per confrontare abbastanza punti a diversi intervalli di distanza è necessario prendere una successione considerevole di scansioni, come viene mostrato in figura 3.12. In questo modo, anche se la visual odometry permette di ottenere pointcloud localmente consistenti, l'errore, accumulato durante il movimento della piattaforma, porta ad ottenere una pointcloud distorta quando vengono considerate diverse scansioni acquisite in intervallo di tempo di qualche secondo. A seguito di diversi tentativi svolti, si è riscontrata una elevata difficoltà nel generare delle mappe la cui proiezione fosse consistente rispetto all'osservatore camera. Di conseguenza, si è deciso di non usare questo dataset per effettuare la validazione della predizione. Si è quindi provato a guardare un successivo lavoro dello stesso gruppo di ricerca: Oxford Robotcar RTK[6] [31]. In questo lavoro viene descritto un approccio che ha permesso di ottenere una ground truth ricostruita utilizzando l'unione dei dati GPS e INS post-processati con le acquisizioni dati della stazione base GNSS (Global Navigation Satellite System) ottenendo una soluzione RTK (Real time Kinematic) globalmente consistente. La stazione base durante tutta la durata dell'acquisizione effettua la correzione dell'er-

[5]https://github.com/ori-mrg/robotcar-dataset-sdk
[6]https://robotcar-dataset.robots.ox.ac.uk/ground_truth/



(a)



(b)

Figura 3.13: Scansioni radar e Velodyne. (a) A sinistra la scansione radar in coordinate polari. A destra la corrispondente scansione in coordinate cartesiane. (b) A sinistra la scansione Velodyne grezza con i valori di intensità in alto e portata in basso. A destra la pointcloud in coordinate cartesiane. Immagini prese da [1].

rore di posizione della piattaforma. Quindi, sono stati utilizzati i dati GPS RTK per poter migliorare la creazione della mappa LIDAR. Tuttavia, anche in questo caso si ottiene una mappa non abbastanza precisa. Infatti, anche se i punti della mappa di profondità proiettati nell'immagine sono allineati correttamente con i rispettivi pixel, l'utilizzo di decine di scansioni monopiano genera nell'immagine molto rumore dovuto alla presenza di ostacoli dinamici nella scena. Anche applicando il filtro di occlusioni questo rumore è difficile da rimuovere. Si è quindi deciso di guardare un ultimo dataset. Oxford Radar RobotCar[7][1] ripercorre gli attraversamenti effettuati da Oxford Robotcar, equipaggiando la piattaforma con due nuovi sensori: due LIDAR Velodyne a 32 piani di scansione, disposti sul tetto ai lati di un radar, in grado di fornire una vista a 360° dell'ambiente circostante durante la navigazione. Un esempio di dati acquisiti da due sensori viene mostrato in figura 3.13.

La presenza di due sensori LIDAR con una maggiore risoluzione permette di migliorare la comprensione 3D della scena rispetto ad utilizzare il LIDAR a 4 piani o le scansioni del LIDAR 2D. Anche in questo caso, l'accuratezza del dato GSP+INS

[7]https://oxford-robotics-institute.github.io/radar-robotcar-dataset/

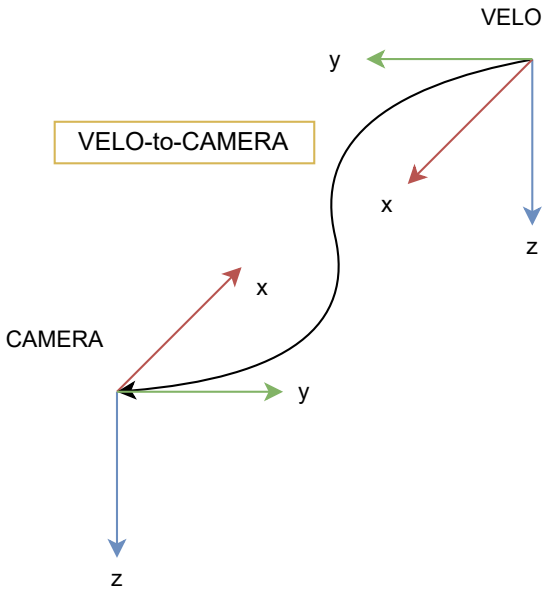


Figura 3.14

varia in modo significativo durante il corso della registrazione dei dati, rendendolo inutilizzabile come ground truth per l'obiettivo della tesi. Di conseguenza la visual odometry rimane la strategia migliore per poter allineare pointcloud localmente consistenti, infatti, maggiore è la qualità dell'allineamento iniziale, maggiore è l'accuratezza del risultato. Per creare la mappa di profondità, inizialmente sono stati utilizzate le acquisizioni di entrambi i sensori Velodyne. Tuttavia, i due sensori LIDAR a 32 piani non sono sincronizzati e di conseguenza la registrazione di più scansioni di entrambi i laser genera una pointcloud distorta. Quindi, si è deciso di utilizzare solo le scansioni acquisite da uno dei due sensori, in particolare quello di sinistra. Come in KITTI, anche in Oxford Robotcar Radar è necessario effettuare le trasformazioni tra sistemi di riferimento per poter prima allineare le scansioni usando la visual odometry e poi applicare l'algoritmo di registrazione. La figura 3.14 rappresenta tutte le trasformazioni necessarie.

La visual odometry viene calcolata rispetto al sistema di riferimento della camera di sinistra. Di conseguenza, per esprimere le coordinate VELO nel sistema di posizione è necessario effettuare solo una roto-traslazione, ovvero VELO_to_CAMERA. Prima di effettuare la registrazione , in ogni pointcloud vengono rimossi i punti che appartengono alla piattaforma, in un intorno di 3.0 metri dal centro del sistema di riferimento, evitando di ottenere una pointcloud mostrata in figura 3.15, dove i punti di colore arancione rappresentano il veicolo ai diversi istanti temporali.

Successivamente, con lo stesso algoritmo utilizzato in KITTI, vengono concatenate le scansioni allineando in modo preciso i punti. Tuttavia, poiché in Oxford le

Figura 3.15: Esempio di allineamenti multipli di scansioni, del dataset Oxford Robotcar Radar, nelle quali non vengono eliminati i punti della piattaforma (in arancione).

Figura 3.16: Esempi di "strisciate" nel dataset Oxford Robotcar, evidenziate nei riquadri rossi.

scene sono molto più dinamiche rispetto a KITTI, anche prendendo poche scansioni rimangono delle "strisciate" causate da ostacoli dinamici che non è possibile rimuovere in modo efficace con il filtro di occlusioni. Si ricorda che il problema è noto ma non viene affrontato in questo lavoro di tesi. Alcuni esempi di questo problema sono mostrati in figura 3.16.

Preprocessing delle immagini

Prima di poter dare in input le immagini alla rete è necessario effettuare del preprocessing specifico per questo dataset. Le immagini rettificate di Oxford hanno una dimensione pari a 1280x960 pixel e contengono al loro interno parte della piattaforma sulla quale sono montate le camere, come si vede in figura 3.17(a). Seguendo l'esempio di KITTI, si è deciso di rimuovere il cofano del veicolo e parti dell'immagi-

(a) Originale.

(b) Rimozione piattaforma.

(c) Rimozione cielo.

(d) Risultato finale.

Figura 3.17: Fasi di ritaglio delle immagini di Oxford Robotcar.

ne su cui non ha senso predire la profondità, come può essere il cielo. In particolare, il processo è stato diviso in 3 fasi:

1. ritaglio della piattaforma dall'immagine (in figura 3.17(b));
2. rimozione delle parti del cielo (in figura 3.17(c));
3. ottenimento del risultato finale applicando un crop centrale (in figura 3.17(d).

L'immagine così ottenuta ha dimensioni pari a 1280x480 e ben rappresenta la scena e gli oggetti all'interno di essa. Questa operazione di preprocessing viene applicata ad ogni immagine in input alla rete durante la fase di training.

Un esempio del risultato della pipeline per la generazione delle mappe LIDAR e la proiezione dei punti nell'immagine processata è mostrato in figura 3.18.

(a) (b) (c)

Figura 3.18: Esempi di immagini di Oxford Robotcar Radar processate utilizzando la nuova pipeline di valutazione. (a) Immagine RGB. (b) Proiezione dei punti 3D della mappa LIDAR nell'immagine. (c) Applicazione del filtro di occlusioni.

Capitolo 4

Valutazione sperimentale

In questo capitolo viene presentata la valutazione sperimentale portata avanti in questo lavoro di tesi. In particolare, nella prima parte viene presentata la configurazione degli esperimenti che viene utilizzata, descrivendo le modalità di training delle reti neurali considerate congiuntamente alle modalità con cui sono state valutate le loro prestazioni. Inoltre, vengono mostrate le diverse configurazioni adottate per poter effettuare i test, variando la soglia di profondità, il numero di pointcloud allineate e nell'intervallo di pointcloud allineate quante tra queste vengono considerate per effettuare la validazione. Nella seconda parte del capitolo vengono mostrati, per ogni dataset e per ogni rete considerata, l'errore ottenuto con le diverse configurazioni e viene effettuata una analisi qualitativa delle mappe di profondità immagine.

4.1 Configurazione degli esperimenti

In questa sezione vengono definite le modalità di training e valutazione e successivamente vengono illustrate le configurazioni per i test con i tre dataset KITTI [13], KITTI-360 [26] e Oxford [30] [1] e con tre reti neurali Monodepth2 [15], Struct2Depth [4] e Depth and motion learning (DML) [25].

4.1.1 Valutazione dell'errore

Il metodo di valutazione che viene utilizzato consiste nel confrontare la mappa di profondità predetta da una rete neurale con la mappa LIDAR ground truth. Con l'obiettivo di valutare e confrontare le performance di varie reti che effettuano stima di profondità, in questo lavoro viene utilizzato il metodo di valutazione proposto da Eigen, Puhrsch e Fergus [10], comunemente utilizzato in letteratura per questo task. Vengono definiti cinque metriche per valutare l'errore rispetto alle predizioni

delle reti: RMSE, RMSE log, Abs Rel, Sq Rel e Accuracies. Questi indicatori sono formulati come segue:

- **RMSE** = $\sqrt{\frac{1}{|N|}\sum_{i \in N} ||d_i - d_i^*||^2}$;
- **RMSE log** = $\sqrt{\frac{1}{|N|}\sum_{i \in N} ||\log(d_i) - \log(d_i^*)||^2}$;
- **Abs Rel** = $\frac{1}{N}\sum_{i \in N} \frac{|d_i - d_i^*|}{d_i^*}$;
- **Sq Rel** = $\frac{1}{N}\sum_{i \in N} \frac{||d_i - d_i^*||^2}{d_i^*}$;
- **Accuracies**: % di d_i tali che $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < threshold$.

dove d_i è il valore di profondità predetto per il pixel i-esimo e d_i^* è il valore ground truth. Inoltre, N è il numero di pixel che viene confrontato tra la GT e la predizione e *threshold* denota il valore di soglia, che in letteratura viene solitamente posto a 1.25. Come in letteratura, per rendere l'ultima metrica meno stringente si considerano anche come valori di soglia 1.25^2 e 1.25^3. RMSE, RMSE log Abs Rel e Sq Rel rappresentano gli errori che si desidera minimizzare, ovvero valori vicini a 0 rappresentano una predizione molto simile alla ground truth. Un'altra strategia di valutazione è quella di esprimere l'accuratezza mediante un valore percentuale, nell'intervallo $[0, 1]$, che rappresenta la quantità di punti predetti che hanno distanza massima dalla ground truth minore della soglia. Di conseguenza, in fase di valutazione le accuratezze rappresentano i valori che si desidera massimizzare, cercando di ottenere valori vicini a 1.

4.1.2 Descrizione della modalità di training

In questo paragrafo viene mostrato come sono state allenate le reti considerate negli esperimenti, con l'obiettivo di ottenere dei risultati comparabili con quelli mostrati in letteratura [15] [4] [25].

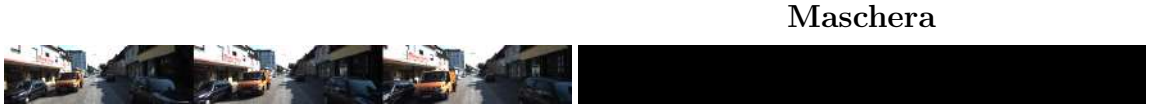
Per ogni dataset si è scelto di utilizzare la stessa dimensione degli esempi in input alle reti, pari a 416x128 pixel. Per mantenere un aspect ratio simile tra l'esempio in input e l'immagine originale, in KITTI è sufficiente applicare un ridimensionamento delle immagini dalla dimensione originale 1242x375 a 416x128. Per Oxford viene effettuato il preprocessing dell'immagine descritto in 3.4.2 con l'obiettivo di rimuovere parti superflue della scena come, ad esempio, il cofano del veicolo utilizzato per la registrazione del dataset e le regioni corrispondenti al cielo. Successivamente, l'immagine ottenuta, di dimensioni pari a 1280x480 pixel, viene ridimensionata a 416x128 pixel. Poiché in questo processo vengono modificate la scala e la dimensione dell'immagine, è necessario successivamente correggere i parametri intrinseci

del modello camera. In particolare, riguardo al dataset di KITTI, viene applicata solo l'operazione di *scaling* in quanto non è necessario rimuovere il cofano e non si hanno tante parti del cielo, quindi, viene adattato il modello camera andando a modificare la lunghezza focale. Invece, per quanto riguarda il dataset di Oxford Robotcar viene ritagliata anche parte dell'immagine effettuando un *crop* centrale del 50% dell'altezza che modifica anche l'origine del piano immagine uv.

Successivamente vengono definiti gli split di training di entrambi i dataset di Oxford Robotcar e KITTI. Nel primo caso, inizialmente sono state sperimentate diverse combinazioni di split che consistono nell'utilizzare immagini prese da una sola sequenza o raggruppando immagini di sequenze diverse. Effettuando diverse prove si è notato come la creazione di un dataset di training composta da immagini prese da sequenze diverse porti ad ottenere una migliore convergenza della funzione di loss. Di conseguenza, si è deciso di creare lo split di training a partire da 50 diverse sequenze degli attraversamenti della città, in modo tale da avere una grande varietà di scene eterogenee. Vengono quindi selezionate all'interno delle 50 sequenze 58830 immagini, di cui 49926 vengono usate per il training e 8904 per la validazione. Nel secondo dataset, invece, viene utilizzato lo split definito da Eigen, Puhrsch e Fergus [10] contenente 22600 coppie di immagini stereo che rappresentano 56 scene etichettate con le categorie "city", "residential" e "road". Di queste 56 scene, 28 vengono utlizzate per la fase di training e le rimanenti per quella di testing. Seguendo il preprocessing per rimuovere i frame statici proposto da Zhou et al. [49], a partire da 45200 immagini si ottengono 39810 immagini per il training e 4424 per la validazione. Di seguito vengono descritti i passaggi effettuati per ogni rete per la fase di training, spiegando su quali dataset sono state allenate dagli autori e su quali invece sono state allenate in questo lavoro. Inoltre, vengono descritte delle modifiche alla rete originale se sono state effettuate.

Monodepth2

La prima rete considerata è Monodepth2 [15]. Godard et al. hanno effettuato i loro esperimenti sui dataset di KITTI e Make3D. In questo lavoro, la rete è stata allenata su KITTI e su Oxford Robotcar utilizzando come punto di partenza i pesi allenati con il dataset di ImageNet. Poiché per allenare Monodepth2 è necessario fornire i parametri intrinseci del modello camera, è stato necessario fornire in input i parametri sia di KITTI che di Oxford Robotcar. Non sono state apportate modifiche all'architettura di rete o ai diversi iperparametri proposti nell'articolo originale. In fase di test la rete genera una mappa di disparità della scena che viene



(a) (b)

Figura 4.1: Esempio in input alla rete Struct2Depth durante la fase di training. (a) Tripletta di immagini RGB. (b) Tripletta di maschere di segmentazione vuote.

successivamente convertita in metri, effettuando uno scaling, per ottenere la mappa di profondità da usare per la valutazione.

Struct2Depth

Struct2Depth [4] è stata allenata da Casser et al. sui dataset di KITTI e Cityscapes. In questo lavoro di tesi la rete è stata addestrata su KITTI e Oxford Robotcar, avendo noti i parametri intrinseci. L'input alla rete è formato da una tripletta di immagini e una tripletta di maschere di segmentazione. Poiché, per il calcolo della profondità non è necessario generare la segmentazione, gli esempi di training contengono al loro interno delle maschere nere, come si può vedere in figura 4.1, sia per il dataset di KITTI che per Oxford. All'inizio del training la rete viene configurata con i pesi già allenati di ResNet. Anche in questo caso non sono state apportate modifiche all'architettura di rete o ai diversi iperparametri proposti nell'articolo originale. Anche questa rete ritorna in output la mappa di profondità scalata in metri.

Depth and motion learning (DML)

DML [25] è stata allenata da Li et al. su KITTI, Cityscapes, Waymo e su delle sequenze video Youtube, in quanto la rete è in grado di stimare anche i parametri intrinseci direttamente dalla sequenza di input. In questo lavoro, la rete è stata allenata solo su KITTI e Oxford Robotcar e i parametri intrinseci vengono considerati noti. Come in Struct2Depth anche in Depth and motion Learning l'input alla rete è formato da triplette di immagini RGB e maschere di segmentazione. Inoltre, anche in questo caso per la fase di training la rete viene configurata con i pesi già allenati di ResNet. Anche per questa rete neurale, non vengono apportate modifiche ad architettura e iperparametri rispetto a quanto proposto dagli autori. L'output della fase di inferenza è una mappa di profondità scalata in metri.

In figura 4.2 vengono mostrati i confronti tra gli errori descritti negli articoli e gli errori delle reti allenate durante il lavoro di tesi. Viene considerata una sola

scansione LIDAR e l'intervallo di profondità è impostato a 80m. Si può vedere come Monodepth2 (RT) abbia dei risultati paragonabili a quelli orginali, Struct2Depth (RT) ottiene dei risultati di poco peggiori, mentre DML ha un notevole peggioramento delle prestazioni. Poichè i dataset KITTI-360 e Oxford Robotcar non sono stati considerati dagli autori per la valutazione, non è stato possibile effettuare un confronto tra gli errori.

4.1.3 Configurazioni

Per valutare in modo efficace il nuovo approccio, si è deciso di eseguire dei test considerando diverse configurazioni:

- diversi intervalli di allineamento;
- diverso numero di pointcloud considerate;
- diversi intervalli di profondità (in metri).

Poiché la metodologia che è stata sviluppata durante il lavoro di tesi rappresenta un'innovazione in letteratura. Per prima cosa si è pensato come effettuare i test e quali intervalli di allineamento considerare. Questo parametro è importante da definire in quanto in base alla quantità di scene statiche e dinamiche presenti nei dataset, l'intervallo di allineamento può essere più o meno grande. In Oxford e anche in KITTI sono stati considerati tre intervalli: 5, 10 e 15 pointcloud allineate. Generalmente, prendere più di 15 scansioni porta ad ottenere un considerevole rumore, quando successivamente viene proiettata la mappa nel piano immagine, dovuto alla presenza di ostacoli dinamici nella scena. In KITTI-360, la sequenza considerata contiene per lo più scene statiche e, di conseguenza, si è deciso di includere un numero maggiore di pointcloud nell'intervallo (ad esempio comprendendo anche 20 e 25 scansioni allineate). Fino ad ora, le scansioni sono state allineate in una successione temporale senza "buchi" di istanti di tempo. Questo metodo che, consiste nel prendere tutte le pointcloud all'interno di uno specifico intervallo, genera una mappa LIDAR nella quale la maggior parte dei punti proiettati è a pochi metri di distanza, in quanto lo spostamento tra le pose delle scansioni è molto piccolo. Di conseguenza, si è pensato di scegliere le pointcloud intervallandole, così che siano presenti, in percentuale, più punti anche a media distanza. Con un intervallo di aggregazione pari a 2 viene selezionata per la mappa LIDAR una pointcloud ogni 2, dimezzando il numero di pointcloud prese per ogni intervallo considerato (ad esempio in un intervallo di 15 pointcloud allineate ne vengono prese 8). Infine vengono definiti tre intervalli di profondità (*Cut*), rispettivamente 30, 50 e 80 metri per vedere come si

comportano le reti e quanto bene riescono a stimare la tridimensionalità della scena a diverse distanze. In tabella 4.1 vengono riassunte le configurazioni di test per ogni dataset.

Dataset	Allineamenti	Cut (in m)	Int. di aggregazione
Oxford	5	30/50/80	2
	10	30/50/80	2
	15	30/50/80	2
KITTI	5	30/50/80	2
	10	30/50/80	2
	15	30/50/80	2
KITTI-360	5	30/50/80	2
	10	30/50/80	2
	15	30/50/80	2
	20	30/50/80	2
	25	30/50/80	2

Tabella 4.1: Tabella riassuntiva delle configurazioni degli esperimenti.

4.2 Risultati

In questa sezione vengono descritti i risultati ottenuti rispetto ai vari test effettuati. Per ogni dataset vengono analizzati i risultati delle reti applicando le metriche per la valutazione nelle diverse configurazioni e confrontando gli errori ottenuti dalle varie reti rispetto alla ground truth basata sull'allineamento di mappe LIDAR. Vengono mostrate le distribuzioni dei punti in base al tipo di allineamento e intervallo di aggregazione utilizzato. Infine, viene effettuata una analisi qualitativa delle mappe di profondità ottenute, mostrando il comportamento delle reti in diversi contesti urbani.

4.2.1 Analisi quantitativa

In questo parte vengono mostrati i risultati ottenuti dal confronto tra la profondità predetta dai modelli considerati e le ground truth con diversi allineamenti. In particolare tale valutazione è stata portata avanti considerando le metriche descritte in 4.1.1. Per ogni dataset vengono mostrate le tabelle che rappresentano l'errore delle

reti a diversi intervalli di profondità. Inoltre, vengono illustrati i grafici di distribuzioni dei punti confrontando per ogni dataset le ground truth dei diversi allineamenti e successivamente vengono confrontate, per ogni intervallo di allineamento, la ground truth e le relative predizioni.

KITTI

Per la fase di test di KITTI viene utilizzato lo split definito da Eigen, Puhrsch e Fergus [10], contenente 697 coppie di immagini. Vengono considerati solo i frame acquisiti dalla camera di sinistra e, quindi, per ogni rete vengono eseguite 697 predizioni. Poiché, la valutazione avviene applicando tre diversi intervalli di allineamenti, si è deciso di considerare tra i 697 frame della camera di sinistra il sottoinsieme contenete le immagini acquisite in un istante temporale con almeno 15 scansioni lidar successive, così da poter effettuare l'allineamento di almeno 15 scansioni. L'insieme ottenuto è composto da 636 immagini. In tabella 4.3 vengono mostrati gli errori per il primo allineamento. Si può notare come al variare della distanza Monodepth2 rimane la rete che ottiene i risultati migliori, seguita da Struct2Depth e DML. All'aumentare del numero di allineamenti il trend non cambia, infatti anche in tabella 4.4 e 4.5 Monodepth2 rimane la rete che ottiene le prestazioni migliori, riuscendo a mantenere l'errore sempre abbastanza contenuto anche con 15 allineamenti a 80 metri di profondità. Il modello che ottiene il secondo miglior risultato è Struct2Depth, che riesce comunque ad ottenere degli errori che non si discostano di molto dai migliori. Invece nel caso del modello Depth and Motion Learning, si riscontra un generale abbassamento delle performance dopo l'applicazione del nuovo metodo di valutazione. Per quanto riguarda invece la distribuzione dei punti al variare della distanza in figura 4.2 vengono mostrati i risultati ottenuti, espressi in percentuale. 4.2(a) rappresenta il confronto delle distribuzioni delle mappe LIDAR, si può vedere come all'aumentare degli allineamenti aumenta anche il numero di punti a distanze maggiori. I restanti grafici 4.2(b-d) convalidano ciò che viene mostrato nelle tabelle 4.3 4.4 4.5, ovvero, al variare degli allineamenti Monodepth2 e Struct2Depth mantengono una distribuzione simile alla GT, mentre DML non riesce a stimare bene la distanza e ciò viene evidenziato dal fatto che nei grafici la distribuzione è poco conforme alla GT. Infine, si può dire che la valutazione con l'applicazione della nuova metodologia, rispetto alla letteratura, mostrata in figura 4.2 porta ad ottenere risultati paragonabili con 5 e 10 allineamenti, mostrando come un maggiore numero di punti non peggiora le performance delle reti. Tuttavia, prendendo 15 scansioni, i risultati hanno un calo di performance ma ciò può essere anche dato dal fatto che si hanno molti più punti a media e lunga distanza.

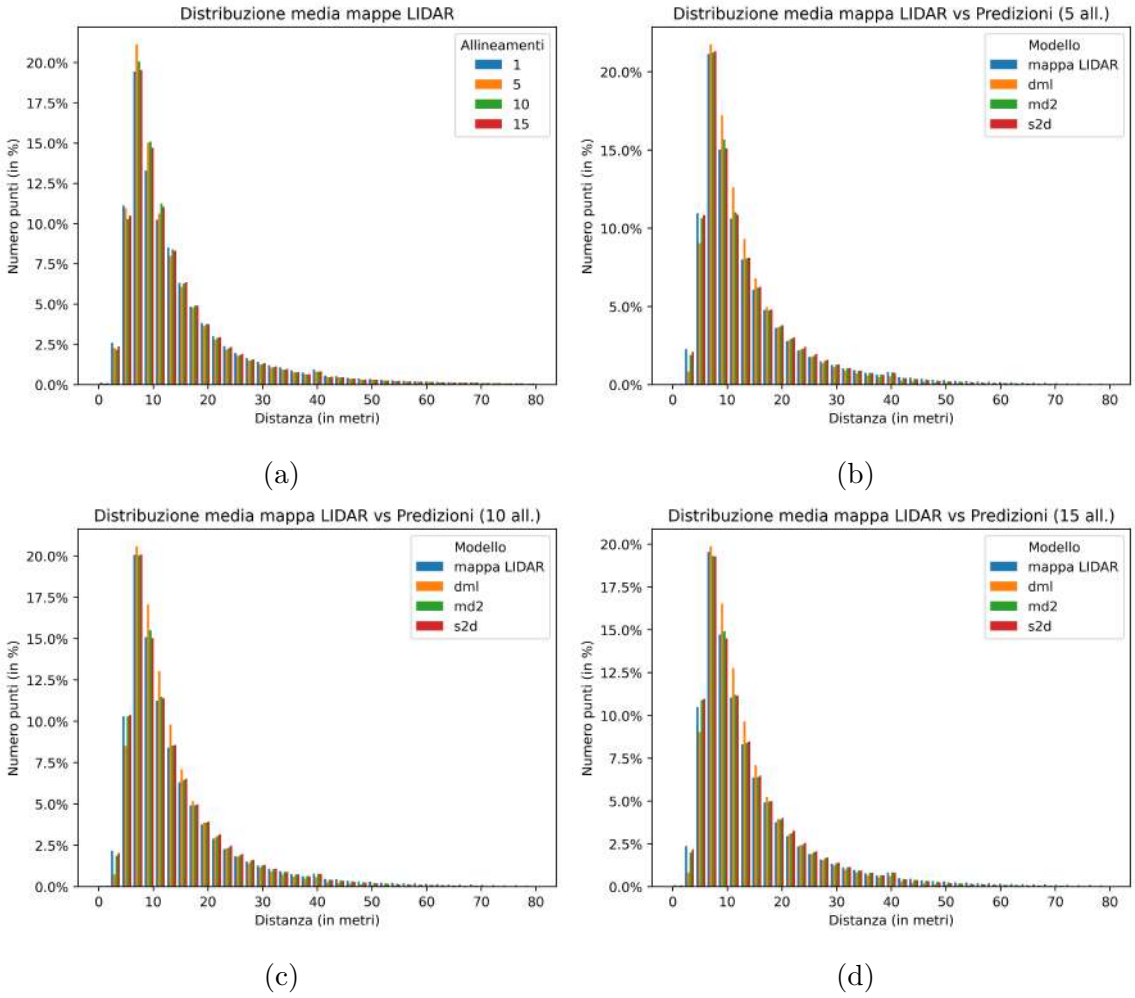


Figura 4.2: Confronti delle distribuzioni medie in percentuale tra le mappe LIDAR (a) o tra mappa LIDAR e le predizioni (b-d) per diversi allineamenti.

KITTI - Valutazione con mappa LIDAR singola scansione									
Metodo	**Train**	**Cap**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	**0.104**	**0.367**	**2.230**	**0.160**	**0.896**	**0.972**	**0.989**
Struct2Depth	M	30m	0.136	0.595	2.622	0.195	0.844	0.955	0.981
DML	M	30m	0.175	0.740	3.100	0.232	0.757	0.924	0.975
Monodepth2	M	50m	**0.119**	**0.671**	**3.543**	**0.187**	**0.869**	**0.961**	**0.984**
Struct2Depth	M	50m	0.151	0.973	4.008	0.221	0.814	0.941	0.975
DML	M	50m	0.191	1.136	4.735	0.262	0.720	0.902	0.964
Monodepth2 (RT)	M	80m	**0.127**	**0.939**	**4.704**	**0.199**	**0.855**	**0.955**	**0.981**
Monodepth2	M	80m	0.128	1.087	5.171	0.204	**0.855**	0.953	0.978
Struct2Depth (RT)	M	80m	0.160	1.311	5.270	0.234	0.799	0.932	0.971
Struct2Depth	M	80m	0.141	1.026	5.291	0.215	0.816	0.945	0.979
DML (RT)	M	80m	0.198	1.417	6.202	0.278	0.704	0.890	0.956
DML	M	80m	0.130	0.950	5.138	0.209	0.843	0.948	0.978

Tabella 4.2: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 1 scansione del dataset KITTI. Inoltre, viene effettuato il confronto tra i risultati dei metodi allenati in questo lavoro di tesi (RT) e i risultati degli articoli originali. In verde vengono rappresentati i valori migliori, sottolineandoli se sono comuni a più reti. Per le metriche in arancione, minori sono i valori migliori sono le performance mentre per quelle blu, maggiori sono i valori migliori sono le performance.

KITTI - Valutazione con mappa LIDAR di 5 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.098**	**0.350**	**2.096**	**0.157**	**0.906**	**0.971**	**0.987**
Struct2Depth	M	30m	2	0.132	0.603	2.521	0.194	0.856	0.954	0.979
DML	M	30m	2	0.161	0.677	2.911	0.220	0.786	0.931	0.977
Monodepth2	M	50m	2	**0.111**	**0.633**	**3.276**	**0.180**	**0.883**	**0.963**	**0.983**
Struct2Depth	M	50m	2	0.147	0.962	3.758	0.218	0.829	0.942	0.974
DML	M	50m	2	0.176	1.021	4.393	0.248	0.754	0.911	0.968
Monodepth2	M	80m	2	**0.118**	**0.898**	**4.367**	**0.192**	**0.872**	**0.957**	**0.981**
Struct2Depth	M	80m	2	0.154	1.269	4.923	0.230	0.817	0.935	0.971
DML	M	80m	2	0.182	1.277	5.790	0.263	0.740	0.901	0.962

Tabella 4.3: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 5 scansioni del dataset KITTI.

KITTI-360

Lo split di test di KITTI-360 contiene al suo interno i frame acquisiti dalla camera sinistra nella sequenza *2013_05_28_drive_000_sync*. Sono state ottenute 214 immagini per il test effettuando un campionamento ogni 50 frame tra gli oltre 11000 presenti all'interno della sequenza. Poiché le scene considerate sono prevalentemente statiche, il dataset viene valutato applicando i seguenti allineamenti: 5, 10, 15, 20 e 25 scansioni. Le tabelle 4.7, 4.8, 4.9, 4.10, 4.11 mostrano i risultati per questi allineamenti. Rispetto a KITTI si può vedere un leggero peggioramento degli errori

KITTI - Valutazione con mappa LIDAR di 10 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.102**	**0.403**	**2.151**	**0.162**	**0.900**	**0.968**	**0.986**
Struct2Depth	M	30m	2	0.137	0.659	2.573	0.199	0.851	0.951	0.977
DML	M	30m	2	0.164	0.724	2.924	0.223	0.786	0.929	0.976
Monodepth2	M	50m	2	**0.118**	**0.746**	**3.357**	**0.187**	**0.877**	**0.959**	**0.981**
Struct2Depth	M	50m	2	0.152	1.054	3.815	0.223	0.826	0.938	0.972
DML	M	50m	2	0.180	1.099	4.407	0.250	0.754	0.910	0.966
Monodepth2	M	80m	2	**0.125**	**1.042**	**4.470**	**0.199**	**0.865**	**0.953**	**0.978**
Struct2Depth	M	80m	2	0.160	1.389	4.996	0.235	0.813	0.931	0.969
DML	M	80m	2	0.186	1.360	5.811	0.265	0.741	0.899	0.960

Tabella 4.4: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 10 scansioni del dataset KITTI.

KITTI - Valutazione con mappa LIDAR di 15 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.118**	**0.534**	**2.324**	**0.178**	**0.880**	**0.960**	**0.982**
Struct2Depth	M	30m	2	0.151	0.788	2.729	0.212	0.829	0.942	0.974
DML	M	30m	2	0.178	0.859	3.065	0.236	0.768	0.920	0.971
Monodepth2	M	50m	2	**0.137**	**1.018**	**3.635**	**0.205**	**0.854**	**0.949**	**0.976**
Struct2Depth	M	50m	2	0.170	1.287	4.042	0.238	0.802	0.928	0.967
DML	M	50m	2	0.197	1.331	4.589	0.264	0.735	0.900	0.961
Monodepth2	M	80m	2	**0.147**	**1.460**	**4.860**	**0.218**	**0.841**	**0.943**	**0.973**
Struct2Depth	M	80m	2	0.179	1.724	5.316	0.251	0.789	0.920	0.963
DML	M	80m	2	0.204	1.639	6.074	0.280	0.721	0.888	0.954

Tabella 4.5: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 15 scansioni del dataset KITTI.

considerando 5 e 10 allineamenti. Monodepth2 rimane la rete che ottiene le performance migliori in tutti i test e Struct2Depth ha valori di poco distanti. Inoltre, si può notare che considerando considerando 15 scansioni KITTI-360 ottiene errori mediamente inferiori rispetto a KITTI. Questo fenomeno si verifica poiché KITTI contiene al suo interno scene con diversi oggetti dinamici. Di conseguenza vi è una possibilità maggiore che concatenando più pointcloud si introduca del rumore all'interno della mappa causato dalle "strisciate" . Anche in questo caso, però, DML ottiene i risultati peggiori, misurando costantemente un errore Abs Rel oltre 0.200 da 10 scansioni in poi. Questo trend viene anche dimostrato dai grafici di distribuzione mostrati in figura 4.3. In figura 4.3(a) si vedono chiaramente i vantaggi di effettuare allineamenti multipli. Infatti un allineamento di 20 e 25 scansioni porta ad ottenere un considerevole aumento di punti a media distanza (dai 10 ai 30 metri) e alcuni punti in più a distanze più grandi. Inoltre, si può notare come in 4.3(b-f) Monodepth2 e Struct2Depth abbiano una distribuzione simile alla GT in tutti i test, portando ad ottenere errori piccoli, mentre si nota chiaramente come DML abbia degli spike isolati che sono la causa di un maggiore errore nella valutazione. Infine,

sono state confrontate le valutazioni del metodo presentato in questo lavoro di tesi con la tecnica della letteratura, i cui risultati sono mostrati in figura 4.6. Si può vedere come gli errori del metodo della letteratura siano più alti dei rispettivi errori per ogni confronto che viene effettuato. Ciò porta a pensare che le predizioni su questo dataset siano molto buone e la presenza di una scena statica con pochi punti rumorosi migliora la valutazione a diversi intervalli di distanza.

KITTI-360 - Valutazione con mappa LIDAR singola scansione									
Metodo	**Train**	**Cap**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	**0.119**	**0.385**	**2.199**	**0.187**	**0.869**	**0.962**	**0.982**
Struct2Depth	M	30m	0.132	0.448	2.325	0.196	0.851	0.959	**0.982**
DML	M	30m	0.194	0.788	3.036	0.257	0.720	0.913	0.966
Monodepth2	M	50m	**0.128**	**0.559**	**3.139**	**0.201**	**0.851**	**0.955**	**0.980**
Struct2Depth	M	50m	0.140	0.623	3.306	0.210	0.830	0.951	**0.980**
DML	M	50m	0.205	1.064	4.291	0.276	0.695	0.896	0.959
Monodepth2	M	80m	**0.131**	**0.675**	**3.974**	**0.208**	**0.842**	**0.950**	**0.978**
Struct2Depth	M	80m	0.143	0.760	4.292	0.219	0.821	0.944	0.977
DML	M	80m	0.209	1.252	5.471	0.287	0.686	0.886	0.953

Tabella 4.6: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 1 scansione del dataset KITTI-360.

KITTI-360 - Valutazione con mappa LIDAR di 5 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.113**	**0.357**	**2.068**	**0.183**	**0.879**	**0.962**	**0.981**
Struct2Depth	M	30m	2	0.125	0.419	2.204	0.192	0.863	0.959	**0.981**
DML	M	30m	2	0.183	0.743	2.899	0.247	0.744	0.920	0.970
Monodepth2	M	50m	2	**0.120**	**0.513**	**2.910**	**0.195**	**0.862**	**0.956**	**0.979**
Struct2Depth	M	50m	2	0.132	0.574	3.088	0.204	0.844	0.952	**0.979**
DML	M	50m	2	0.193	0.979	4.027	0.264	0.721	0.905	0.964
Monodepth2	M	80m	2	**0.123**	**0.611**	**3.653**	**0.201**	**0.856**	**0.952**	**0.978**
Struct2Depth	M	80m	2	0.136	0.694	3.984	0.211	0.836	0.947	0.977
DML	M	80m	2	0.197	1.139	5.099	0.274	0.712	0.897	0.959

Tabella 4.7: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 5 scansioni del dataset KITTI-360.

Oxford Robotcar

Il test delle reti allenate su Oxford Robotcar viene effettuato considerando il dataset Oxford Robotcar Radar in quanto contiene il dato LIDAR Velodyne a 32 piani. Lo split di test di Oxford Robotcar Radar contiene al suo interno i frame acquisiti dalla camera sinistra, della configurazione stereo, nella sequenza *2019-01-10-14-36-48*. Tra questi ne sono stati selezionati 75 in modo tale da avere scene statiche ma anche molte scene dinamiche, per poter effettuare una valutazione in condizioni

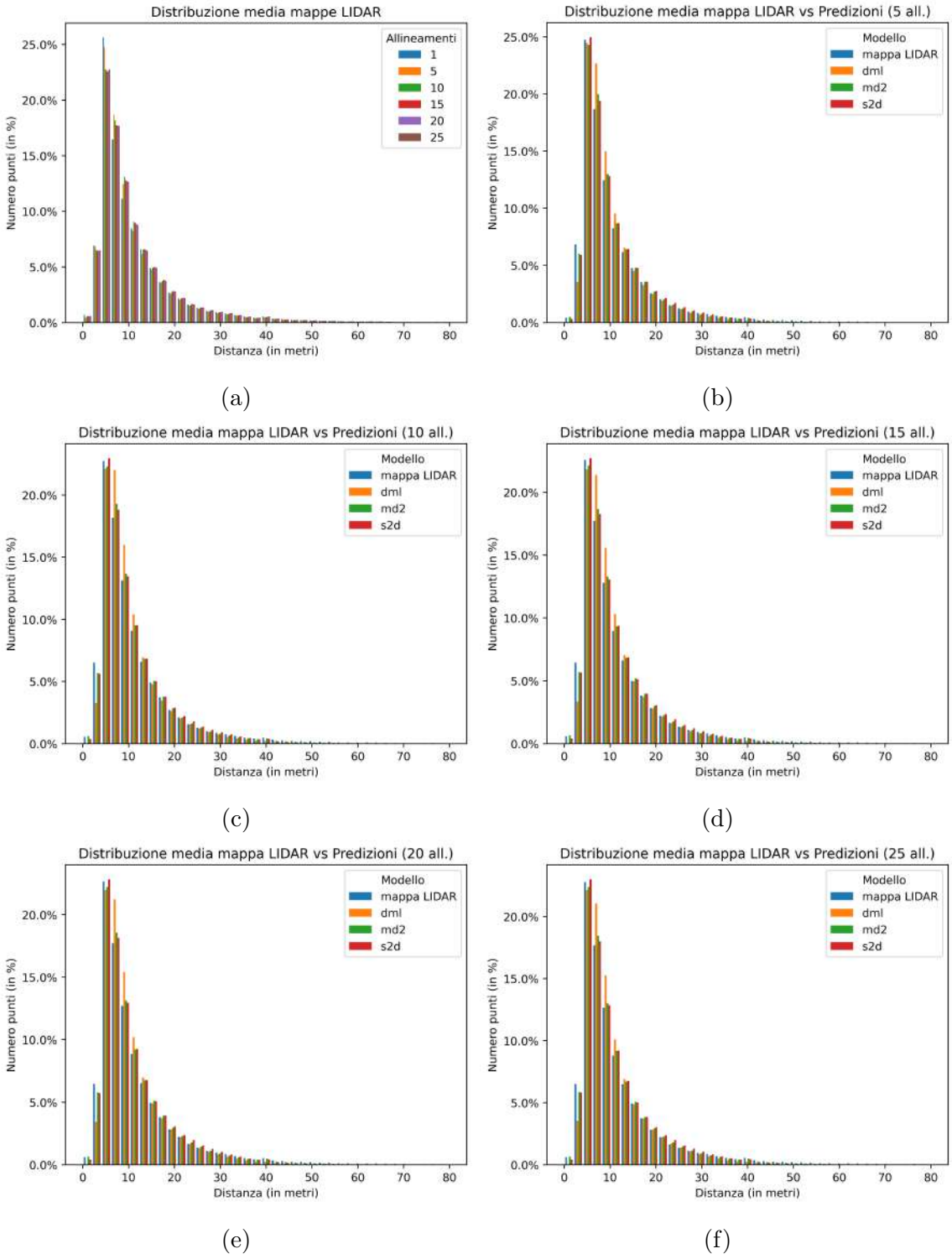


Figura 4.3: Confronti delle distribuzioni medie in percentuale tra le mappe LIDAR (a) o tra mappa LIDAR e le predizioni (b-f) per diversi allineamenti.

KITTI-360 - Valutazione con mappa LIDAR di 10 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.115**	**0.366**	**2.055**	**0.182**	**0.882**	**0.962**	**0.982**
Struct2Depth	M	30m	2	0.127	0.428	2.194	0.191	0.864	0.959	0.981
DML	M	30m	2	0.188	0.768	2.907	0.249	0.741	0.919	0.969
Monodepth2	M	50m	2	**0.122**	**0.524**	**2.893**	**0.194**	**0.866**	**0.956**	**0.980**
Struct2Depth	M	50m	2	0.134	0.587	3.077	0.203	0.845	0.953	**0.980**
DML	M	50m	2	0.197	1.001	4.015	0.266	0.719	0.904	0.963
Monodepth2	M	80m	2	**0.125**	**0.624**	**3.648**	**0.199**	**0.859**	**0.952**	**0.979**
Struct2Depth	M	80m	2	0.137	0.714	4.000	0.210	0.837	0.947	0.978
DML	M	80m	2	0.201	1.164	5.104	0.275	0.711	0.896	0.958

Tabella 4.8: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 10 scansioni del dataset KITTI-360.

KITTI-360 - Valutazione con mappa LIDAR di 15 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.117**	**0.385**	**2.071**	**0.183**	**0.882**	**0.962**	**0.982**
Struct2Depth	M	30m	2	0.129	0.449	2.216	0.192	0.863	0.959	**0.982**
DML	M	30m	2	0.191	0.794	2.950	0.251	0.737	0.918	0.969
Monodepth2	M	50m	2	**0.125**	**0.557**	**2.943**	**0.195**	**0.865**	**0.956**	**0.980**
Struct2Depth	M	50m	2	0.137	0.622	3.139	0.204	0.843	0.952	**0.980**
DML	M	50m	2	0.200	1.036	4.089	0.268	0.714	0.903	0.963
Monodepth2	M	80m	2	**0.128**	**0.668**	**3.745**	**0.201**	**0.858**	**0.952**	**0.979**
Struct2Depth	M	80m	2	0.140	0.763	4.126	0.212	0.835	0.946	0.978
DML	M	80m	2	0.204	1.213	5.253	0.279	0.705	0.894	0.958

Tabella 4.9: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 15 scansioni del dataset KITTI-360.

KITTI-360 - Valutazione con mappa LIDAR di 20 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.118**	**0.399**	**2.086**	**0.184**	**0.880**	**0.961**	**0.982**
Struct2Depth	M	30m	2	0.131	0.464	2.233	0.193	0.863	0.958	**0.982**
DML	M	30m	2	0.193	0.809	2.967	0.253	0.734	0.916	0.968
Monodepth2	M	50m	2	**0.126**	**0.580**	**2.976**	**0.196**	**0.863**	**0.955**	**0.980**
Struct2Depth	M	50m	2	0.139	0.645	3.178	0.206	0.842	0.952	**0.980**
DML	M	50m	2	0.202	1.057	4.132	0.270	0.711	0.901	0.962
Monodepth2	M	80m	2	**0.129**	**0.697**	**3.806**	**0.203**	**0.856**	**0.951**	**0.979**
Struct2Depth	M	80m	2	0.142	0.793	4.202	0.214	0.833	0.945	0.977
DML	M	80m	2	0.206	1.243	5.343	0.281	0.702	0.892	0.956

Tabella 4.10: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 20 scansioni del dataset KITTI-360.

molto complicate, con pedoni, veicoli e altri oggetti che si muovono all'interno della scena in direzioni e a velocità diverse. Quindi si è cercato di valutare il dataset con i seguenti allineamenti: 5, 10 e 15 scansioni. Come mostrato nelle tabelle 4.13, 4.14 e 4.15, in controtendenza rispetto a KITTI e KITTI-360, la rete che ha ottenuto le prestazioni migliori è stata DML. Anche se il suo errore rimane comunque elevato,

KITTI-360 - Valutazione con mappa LIDAR di 25 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	**0.120**	**0.418**	**2.105**	**0.186**	**0.879**	**0.961**	**0.982**
Struct2Depth	M	30m	2	0.133	0.484	2.252	0.195	0.861	0.958	0.981
DML	M	30m	2	0.195	0.827	2.980	0.255	0.732	0.915	0.967
Monodepth2	M	50m	2	**0.128**	**0.608**	**3.012**	**0.198**	**0.862**	**0.954**	**0.980**
Struct2Depth	M	50m	2	0.141	0.675	3.220	0.208	0.840	0.951	0.979
DML	M	50m	2	0.204	1.084	4.170	0.272	0.708	0.900	0.961
Monodepth2	M	80m	2	**0.132**	**0.732**	**3.864**	**0.205**	**0.854**	**0.950**	**0.978**
Struct2Depth	M	80m	2	0.145	0.831	4.276	0.216	0.830	0.944	0.977
DML	M	80m	2	0.208	1.279	5.425	0.283	0.699	0.890	0.955

Tabella 4.11: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 25 scansioni del dataset KITTI-360.

Monodepth2 e Struct2Depth ottengono prestazioni peggiori. Infatti, Monodepth2 che ha ottenuto prestazioni molto buone nelle scene semi statiche di KITTI, ha peggiorato notevolmente le performance nelle scene molto dinamiche del dataset di Oxford Robotcar. Questo risultato viene mostrato anche dai grafici di distribuzione dei punti mostrati in figura 4.4. Come nei test svolti sui dataset di KITTI e KITTI-360, 4.4(a) mostra la distribuzione tra le mappe LIDAR con diversi allineamenti, confermando come si ottengano più punti a medie distanze considerando un intervallo di allineamenti maggiore. Le figure 4.4(b-d) mostrano la distribuzione delle predizioni rispetto alla relativa mappa LIDAR. Si può vedere come tutte le reti non siano in grado di generare una predizione accurata dei punti a distanze piccole, che quindi, essendo una grossa percentuale dei punti, pesano molto sul calcolo dell'errore totale. Infine, viene effettuato il confronto con la singola scansione del metodo della letteratura, la cui tabella è mostrata in 4.12. Gli errori, seppure più alti di KITTI e KITTI-360, rimangono consistenti con la valutazione delle mappe LIDAR allineate. Quindi si può anche evincere che la presenza di "strisciate" causate dai molti ostacoli dinamici influenza poco il risultato finale della valutazione, proprio perché non c'è un netto miglioramento degli errori nel caso della singola scansione.

4.2.2 Analisi qualitativa

In questo ultimo paragrafo viene effettuata una analisi qualitativa, mostrando, per ogni dataset, le immagini delle mappe di profondità ottenute dalle reti. Le immagini sono state prese considerando la soglia massima di profondità impostata a 80 metri.

KITTI

Come si è visto, KITTI contiene al suo interno scene sia statiche che dinamiche. Sono state selezionate alcune tra le 697 immagini testate per poter valutare la qualità

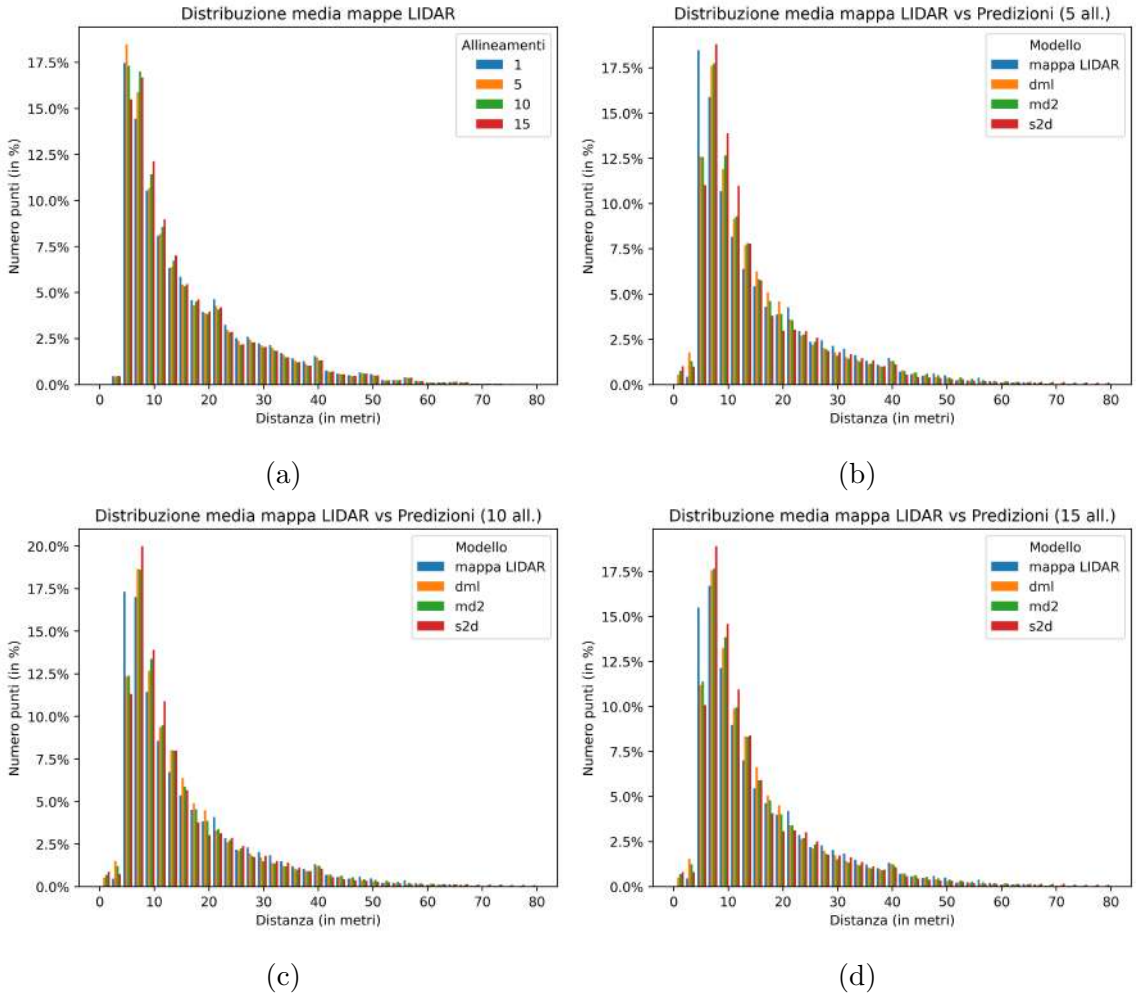


Figura 4.4: Confronti delle distribuzioni medie in percentuale tra le mappe LIDAR (a) o tra mappa LIDAR e le predizioni (b-d) per diversi allineamenti.

OXFORD - Valutazione con mappa LIDAR singola scansione									
Metodo	**Train**	**Cap**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	0.221	1.340	4.025	0.306	0.688	0.864	0.932
Struct2Depth	M	30m	0.230	1.188	4.011	0.370	0.642	0.845	0.917
DML	M	30m	**0.198**	**1.017**	**3.789**	**0.291**	**0.704**	**0.881**	**0.948**
Monodepth2	M	50m	0.267	2.494	6.291	0.355	0.632	0.837	0.912
Struct2Depth	M	50m	0.267	2.011	6.321	0.414	0.585	0.811	0.893
DML	M	50m	**0.238**	**1.862**	**6.152**	**0.340**	**0.640**	**0.847**	**0.925**
Monodepth2	M	80m	0.282	3.199	7.416	0.372	**0.621**	0.830	0.906
Struct2Depth	M	80m	0.284	2.733	7.670	0.433	0.572	0.795	0.885
DML	M	80m	**0.246**	**2.162**	**7.151**	**0.353**	0.620	**0.840**	**0.919**

Tabella 4.12: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 1 scansione del dataset Oxford Robotcar.

OXFORD - Valutazione con mappa LIDAR di 5 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	0.214	1.296	3.918	0.302	0.703	0.866	0.930
Struct2Depth	M	30m	2	0.223	1.148	3.905	0.368	0.660	0.851	0.919
DML	M	30m	2	**0.192**	**0.977**	**3.696**	**0.288**	**0.713**	**0.882**	**0.951**
Monodepth2	M	50m	2	0.260	2.415	6.163	0.352	0.653	0.834	0.912
Struct2Depth	M	50m	2	0.260	1.965	6.209	0.413	0.612	0.811	0.897
DML	M	50m	2	**0.230**	**1.800**	**6.029**	**0.337**	**0.661**	**0.847**	**0.927**
Monodepth2	M	80m	2	0.274	3.111	7.297	0.368	**0.645**	0.826	0.906
Struct2Depth	M	80m	2	0.276	2.673	7.539	0.431	0.598	0.798	0.887
DML	M	80m	2	**0.239**	**2.147**	**7.077**	**0.350**	0.642	**0.838**	**0.921**

Tabella 4.13: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 5 scansioni del dataset Oxford Robotcar.

OXFORD - Valutazione con mappa LIDAR di 10 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	0.219	1.357	3.914	0.304	0.702	0.866	0.928
Struct2Depth	M	30m	2	0.223	1.155	3.872	0.369	0.669	0.855	0.921
DML	M	30m	2	**0.196**	**0.996**	**3.683**	**0.291**	**0.712**	**0.881**	**0.950**
Monodepth2	M	50m	2	0.263	2.515	6.145	0.354	0.652	0.832	0.909
Struct2Depth	M	50m	2	0.259	1.975	6.146	0.414	0.614	0.813	0.899
DML	M	50m	2	**0.233**	**1.824**	**5.999**	**0.340**	**0.654**	**0.842**	**0.926**
Monodepth2	M	80m	2	0.277	3.175	7.273	0.370	**0.639**	0.825	0.903
Struct2Depth	M	80m	2	0.275	2.679	7.483	0.432	0.599	0.802	0.890
DML	M	80m	2	**0.242**	**2.169**	**7.061**	**0.354**	0.638	**0.834**	**0.920**

Tabella 4.14: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 10 scansioni del dataset Oxford Robotcar.

visiva della ricostruzione, mostrando i risultati in figura 4.5. Si può vedere come nelle mappe di profondità ottenute da Monodepth2, in generale, gli oggetti sia statici che dinamici siano ben rappresentati nella scena con i contorni discretamente definiti. Struct2Depth ottiene dei risultati simili a Monodepth2 ma i contorni sono meno definiti e in alcuni esempi si può vedere come il valore di profondità assegnato ai

OXFORD - Valutazione con mappa LIDAR di 15 scansioni										
Metodo	**Train**	**Cap**	**Int Aggr**	**Abs Rel**	**Sq Rel**	**RMSE**	**RMSE log**	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Monodepth2	M	30m	2	0.225	1.411	3.958	0.310	0.692	0.864	0.926
Struct2Depth	M	30m	2	0.225	1.179	3.876	0.372	0.665	0.857	0.921
DML	M	30m	2	**0.202**	**1.023**	**3.709**	**0.296**	**0.701**	**0.879**	**0.949**
Monodepth2	M	50m	2	0.268	2.570	6.201	0.359	0.639	0.830	0.908
Struct2Depth	M	50m	2	0.262	2.009	6.170	0.417	0.612	0.815	0.899
DML	M	50m	2	**0.240**	**1.875**	**6.056**	**0.345**	**0.636**	**0.842**	**0.926**
Monodepth2	M	80m	2	0.281	3.202	7.374	0.375	**0.629**	0.822	0.901
Struct2Depth	M	80m	2	0.279	2.763	7.601	0.437	0.597	0.802	0.889
DML	M	80m	2	**0.249**	**2.242**	**7.181**	**0.360**	0.621	**0.831**	**0.919**

Tabella 4.15: Confronto tra i risultati dei metodi, considerando come GT una mappa LIDAR composta da 15 scansioni del dataset Oxford Robotcar.

pixel sia poco preciso, in quanto si possono notare delle sezioni non omogenee sullo stesso oggetto. DML è la rete che ottiene una valutazione qualitativa peggiore tra le tre considerate. Nelle regioni di discontinuità tra oggetti in primo piano e background i contorni sono poco definiti. In generale però la mappa di profondità non contiene parti disomogenee.

KITTI-360

Le 214 immagini contenute nella sequenza scelta di KITTI-360 contengono al loro interno prevalentemente scene statiche, con alcuni casi di veicoli in movimento. In figura 4.6 vengono mostrati alcuni tra gli esempi testati con le tre reti. Come in KITTI, anche in questo dataset, le mappe di profondità ottenute da Monodepth2 siano ben rappresentanti la scena, con dei netti contorni che definiscono in modo chiaro gli oggetti. Anche le mappe di profondità di Struct2Depth ottengono dei risultati qualitativi molto simili a Monodepth2. Infatti, a differenza dei test effettuati su KITTI, gli oggetti viene definito meglio il contorno degli oggetti e l'immagine non presenta sezioni disomogenee. Anche in questo caso DML ha ottenuto le performance peggiori tra le tre reti. Come in KITTI, anche in KITTI-360, le mappe di profondità sono poco definite ed è difficile distinguere nitidamente gli oggetti all'interno della scena.

Oxford Robotcar

Le reti testate con il dataset di Oxford Robotcar ha ottenuto le prestazioni medie peggiori nell'analisi quantitativa, tra i vari dataset valutati. L'analisi qualitativa viene effettuata considerando le stesse 75 immagini di Oxford Robotcar Radar scelte per il test. In figura 4.7 vengono mostrate alcune tra le immagini testate. Si può vedere come, in questo caso, Monodepth2 generi una mappa di profondità poco

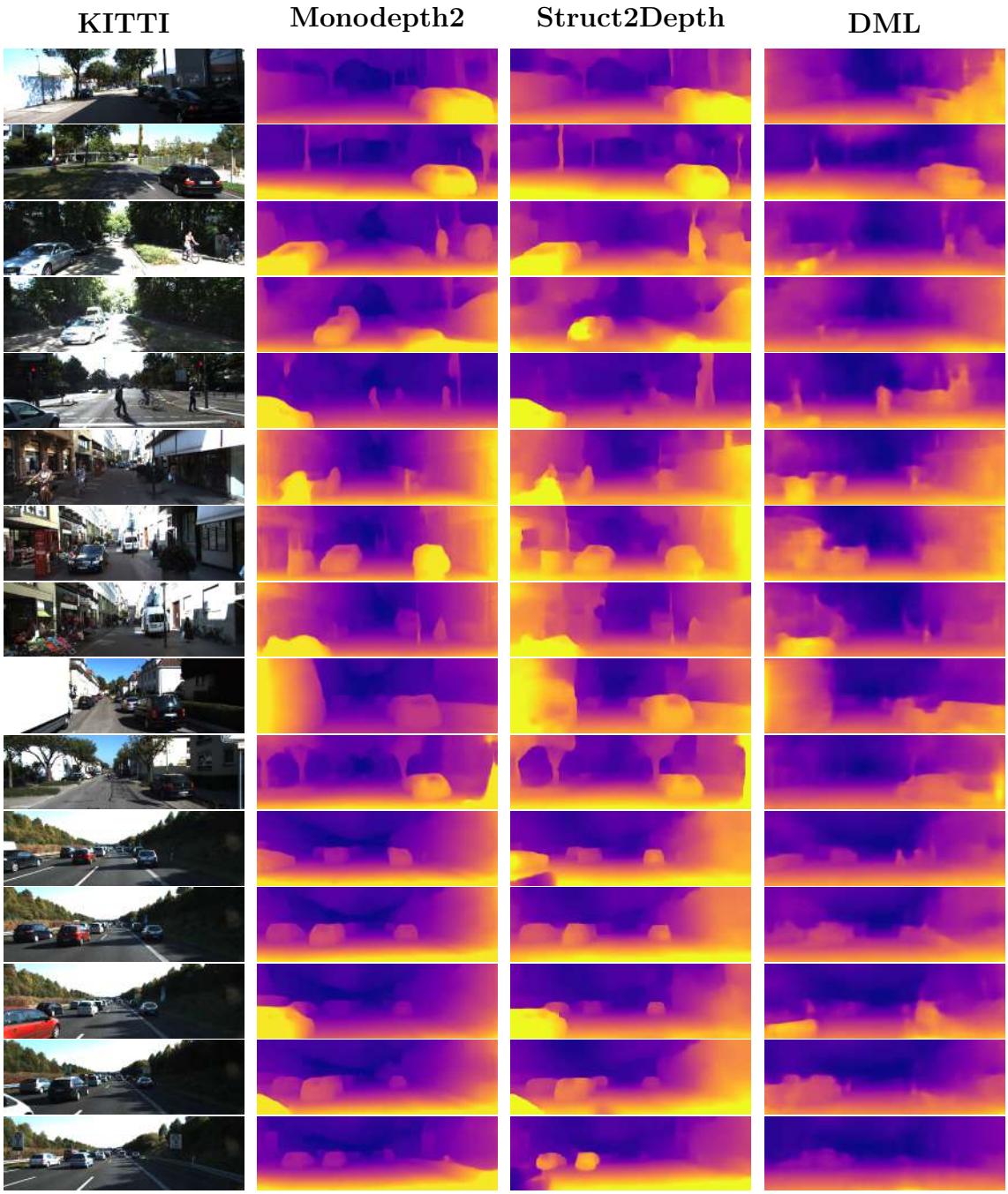


Figura 4.5: Esempi di predizione di profondità delle immagini di KITTI.

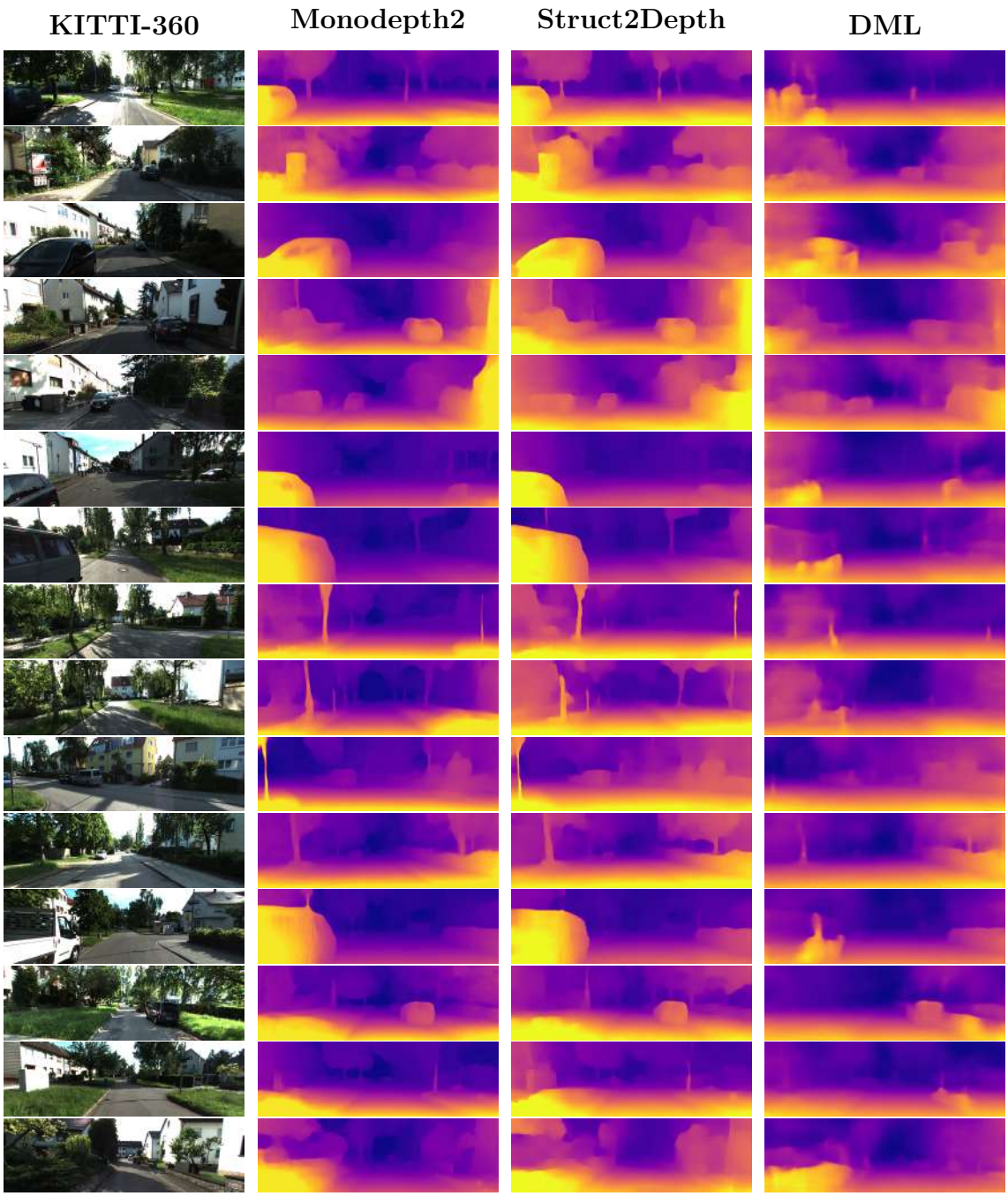


Figura 4.6: Esempi di predizione di profondità delle immagini di KITTI-360.

accurata rispetto all'immagine originale. Nelle immagini molto dinamiche si vede chiaramente la difficoltà nel stimare la profondità, ad esempio non riuscendo a rappresentare correttamente i veicoli in movimento. Struct2Depth ottiene risultati peggiori rispetto a KITTI e KITTI-360 ma riesce meglio rispetto a Monodepth2 a rappresentare all'interno della scena gli oggetti dinamici. DML ha ottenuto le prestazioni migliori nell'analisi quantitativa. Dalle immagini si può notare infatti come, anche se con una grande difficoltà, la profondità venga stimata meglio, e in alcuni casi di veicoli in movimento i confini vengono rappresentati abbastanza nitidamente. Tuttavia, sono anche presenti porzioni di immagine dove si ha un valore di profondità molto distante dal dato di ground truth.

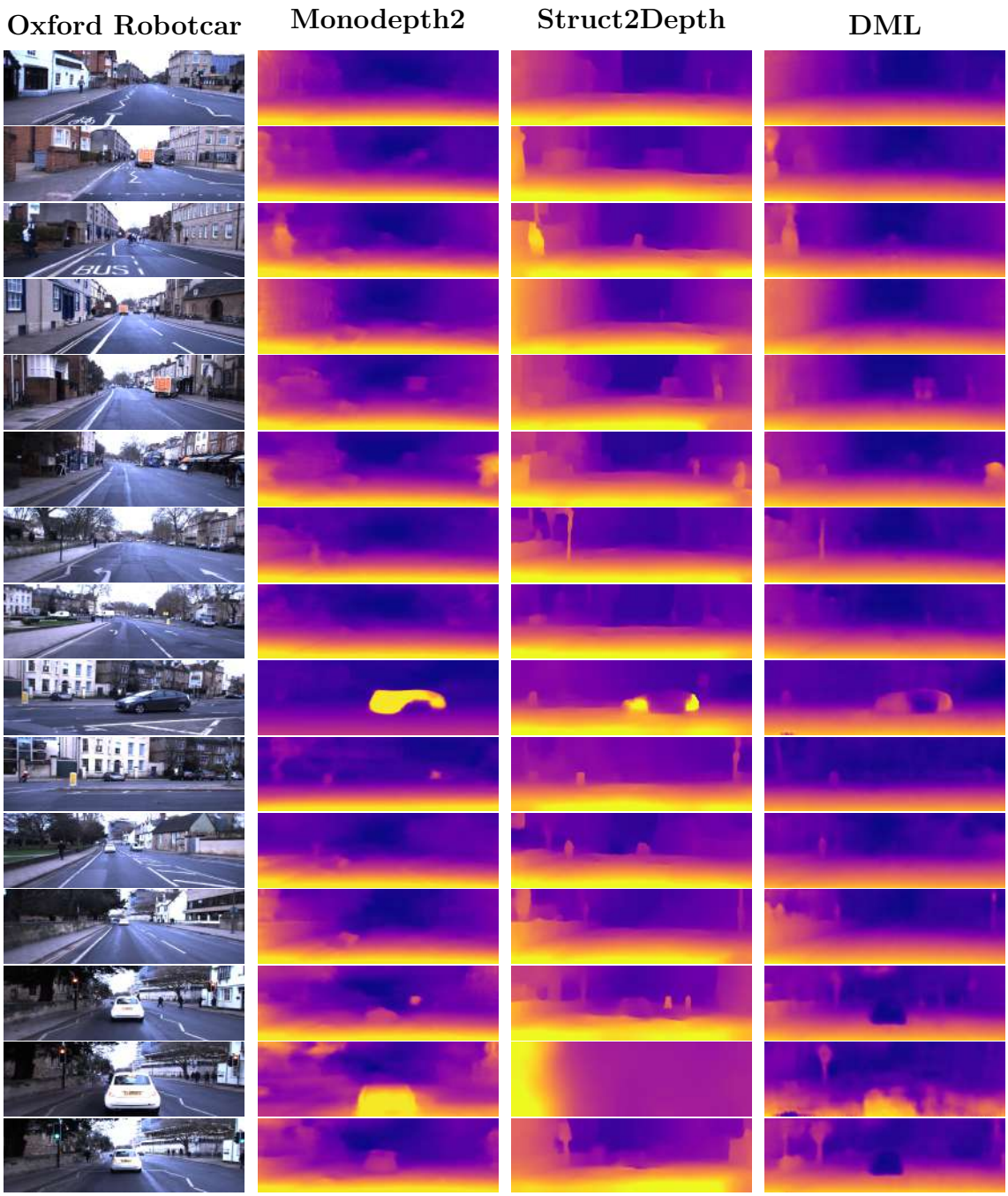


Figura 4.7: Esempi di predizione di profondità delle immagini di Oxford Robotcar.

Capitolo 5

Conclusioni

In questo lavoro di tesi è stato presentato un nuovo metodo di valutazione delle prestazioni di reti neurali che stimano la profondità di una scena avendo come informazione in input solo una singola immagine. In particolare, la principale distinzione rispetto alle tecniche presenti in letteratura, basate sul confronto tra la proiezione di una scansione e la predizione della rete, è l'utilizzo di mappe di profondità generate dall'allineamento di più scansioni LIDAR.

Oggigiorno, il task di stima di profondità da singola immagine è di massima attualità in quanto, in ambito automotive, l'utilizzo di camere permette di mantenere i costi bassi, avere un notevole campo di misurazione e al contempo essere più affidabili di altri sensori come i LIDAR che, avendo componenti meccaniche, sono poco affidabili e durevoli nelle condizioni di lavoro di un autoveicolo. L'approccio sperimentale sviluppato in questo lavoro di tesi permette di valutare meglio la stima di profondità di reti esistenti in letteratura.

Durante il lavoro di tesi ci si è concentrati nel creare mappe di profondità dense allineando scansioni LIDAR, utilizzando la tecnica di *pointcloud registration*, in modo tale che la mappa generata possa essere proiettata in un piano immagine virtuale con una corrispondenza accurata tra il valore del punto e il rispettivo pixel immagine. Per ridurre il rumore causato dalla proiezione di punti di scansioni prese a pose e istanti temporali diversi, nell'immagine è stato applicato un filtro in grado di rimuovere i punti della mappa LIDAR che risultano occlusi rispetto al punti di vista della camera. Tuttavia, non viene effettuata alcuna operazione di rimozione di "strisciate" di punti causate dal movimento degli ostacoli dinamici.

I risultati, ottenuti dall'applicazione di reti neurali su diversi dataset, mostrano come il lavoro svolto possa considerarsi un buon punto di partenza per affrontare il problema della valutazione di stima della profondità. I test effettuati mostrano come le reti neurali selezionate ottengano risultati diversi al variare della tipologia di

scene urbane considerate e del numero di scansioni LIDAR prese in considerazione per la valutazione.

Si ritiene che la nuova metrica presentata nel lavoro di tesi abbia buone potenzialità e che, con delle opportune modifiche, possa essere utilizzata per migliorare la valutazione degli approcci presenti nello stato dell'arte.

Di seguito vengono elencate alcune tra le possibili migliorie che possono essere applicate al lavoro:

- implementazione di un metodo di segmentazione semantica sia di pointcloud che di immagini in modo tale da riconoscere e rimuovere le "strisciate", causate dagli oggetti dinamici;
- estensione del metodo di valutazione a un maggior numero di reti neurali e a nuovi dataset, come possono essere Waymo, Argoverse o Lyft;
- estensione del metodo di valutazione a metodi supervisionati;
- effettuare test con diversi intervalli di allineamento e aggregazione di pointcloud per poter confrontare i risultati esposti nel lavoro di tesi.

Bibliografia

[1] Dan Barnes et al. «The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset». In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6433–6438.

[2] Paul J Besl e Neil D McKay. «Method for registration of 3-D shapes». In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[3] Duane C Brown. «Decentering distortion of lenses». In: *Photogrammetric Engineering and Remote Sensing* (1966).

[4] Vincent Casser et al. «Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos». In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 8001–8008.

[5] Daniele Cattaneo et al. «Cmrnet: Camera to lidar-map registration». In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 1283–1289.

[6] Yang Chen e Gérard Medioni. «Object modelling by registration of multiple range images». In: *Image and vision computing* 10.3 (1992), pp. 145–155.

[7] AE Conrady. «Lens-systems, decentered». In: *Monthly notices of the royal astronomical society* 79 (1919), pp. 384–390.

[8] Marius Cordts et al. «The cityscapes dataset for semantic urban scene understanding». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[9] David Eigen e Rob Fergus. «Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture». In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2650–2658.

[10] David Eigen, Christian Puhrsch e Rob Fergus. «Depth map prediction from a single image using a multi-scale deep network». In: *Advances in neural information processing systems* 27 (2014).

[11] Huan Fu et al. «Deep ordinal regression network for monocular depth estimation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2002–2011.

[12] Andrea Fusiello, Emanuele Trucco e Alessandro Verri. «A compact algorithm for rectification of stereo pairs». In: *Machine vision and applications* 12.1 (2000), pp. 16–22.

[13] Andreas Geiger et al. «Vision meets robotics: The kitti dataset». In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[14] Clément Godard, Oisin Mac Aodha e Gabriel J Brostow. «Unsupervised monocular depth estimation with left-right consistency». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279.

[15] Clément Godard et al. «Digging into self-supervised monocular depth estimation». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3828–3838.

[16] Jonas Gomes e Luiz Velho. *Image processing for computer graphics*. Springer Science & Business Media, 1997.

[17] Ariel Gordon et al. «Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8977–8986.

[18] Kaiming He et al. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[19] Armin Hornung et al. «OctoMap: An efficient probabilistic 3D mapping framework based on octrees». In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[20] Kevin Karsch, Ce Liu e Sing Bing Kang. «Depth transfer: Depth extraction from video using non-parametric sampling». In: *IEEE transactions on pattern analysis and machine intelligence* 36.11 (2014), pp. 2144–2158.

[21] Lubor Ladicky, Jianbo Shi e Marc Pollefeys. «Pulling things out of perspective». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 89–96.

[22] John Lafferty, Andrew McCallum e Fernando CN Pereira. «Conditional random fields: Probabilistic models for segmenting and labeling sequence data». In: (2001).

[23] Iro Laina et al. «Deeper depth prediction with fully convolutional residual networks». In: *2016 Fourth international conference on 3D vision (3DV)*. IEEE. 2016, pp. 239–248.

[24] Bo Li et al. «Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1119–1127.

[25] Hanhan Li et al. «Unsupervised monocular depth learning in dynamic scenes». In: *arXiv preprint arXiv:2010.16404* (2020).

[26] Yiyi Liao, Jun Xie e Andreas Geiger. «KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d». In: *arXiv preprint arXiv:2109.13410* (2021).

[27] Fayao Liu et al. «Learning depth from single monocular images using deep convolutional neural fields». In: *IEEE transactions on pattern analysis and machine intelligence* 38.10 (2015), pp. 2024–2039.

[28] Chenxu Luo et al. «Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding». In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2624–2641.

[29] Haoran Lyu et al. «Esnet: Edge-based segmentation network for real-time semantic segmentation in traffic scenes». In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 1855–1859.

[30] Will Maddern et al. «1 year, 1000 km: The Oxford RobotCar dataset». In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15.

[31] Will Maddern et al. «Real-time kinematic ground truth for the oxford robotcar dataset». In: *arXiv preprint arXiv:2002.10152* (2020).

[32] Nikolaus Mayer et al. «A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.

[33] Daniel Oram. «Rectification for any epipolar geometry.» In: *BMVC*. Vol. 1. Citeseer. 2001, pp. 653–662.

[34] Jiangmiao Pang et al. «Libra r-cnn: Towards balanced learning for object detection». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 821–830.

[35] Patrick Pfaff, Rudolph Triebel e Wolfram Burgard. «An efficient extension to elevation maps for outdoor terrain mapping and loop closing». In: *The International Journal of Robotics Research* 26.2 (2007), pp. 217–230.

[36] Ruggero Pintus, Enrico Gobbetti e Marco Agus. «Real-time rendering of massive unstructured raw point clouds using screen-space operators». In: *Proceedings of the 12th International conference on Virtual Reality, Archaeology and Cultural Heritage*. 2011, pp. 105–112.

[37] Marc Pollefeys, Reinhard Koch e Luc Van Gool. «A simple and efficient rectification method for general motion». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. IEEE. 1999, pp. 496–501.

[38] Ashutosh Saxena, Min Sun e Andrew Y Ng. «Make3d: Learning 3d scene structure from a single still image». In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 824–840.

[39] Daniel Scharstein e Richard Szeliski. «A taxonomy and evaluation of dense two-frame stereo correspondence algorithms». In: *International journal of computer vision* 47.1 (2002), pp. 7–42.

[40] Aleksandr Segal, Dirk Haehnel e Sebastian Thrun. «Generalized-icp.» In: *Robotics: science and systems*. Vol. 2. 4. Seattle, WA. 2009, p. 435.

[41] Nathan Silberman et al. «Indoor segmentation and support inference from rgbd images». In: *European conference on computer vision*. Springer. 2012, pp. 746–760.

[42] Rudolph Triebel, Patrick Pfaff e Wolfram Burgard. «Multi-level surface maps for outdoor terrain mapping and loop closing». In: *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2006, pp. 2276–2282.

[43] Emanuele Trucco e Alessandro Verri. *Introductory techniques for 3-D computer vision*. Vol. 201. Prentice hall Englewood Cliffs, 1998.

[44] Shimon Ullman. «The interpretation of structure from motion». In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 203.1153 (1979), pp. 405–426.

[45] Chaoyang Wang et al. «Learning depth from monocular videos using direct methods». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2022–2030.

[46] Feng Zhang, Xiatian Zhu e Mao Ye. «Fast human pose estimation». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3517–3526.

[47] Zhengyou Zhang. «Iterative point matching for registration of free-form curves and surfaces». In: *International journal of computer vision* 13.2 (1994), pp. 119–152.

[48] Chaoqiang Zhao et al. «Monocular depth estimation based on deep learning: An overview». In: *Science China Technological Sciences* 63.9 (2020), pp. 1612–1627.

[49] Tinghui Zhou et al. «Unsupervised learning of depth and ego-motion from video». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1851–1858.

[50] Laurent Zwald e Sophie Lambert-Lacroix. «The berhu penalty and the grouped effect». In: *arXiv preprint arXiv:1207.6868* (2012).