UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in Ingegneria Informatica

Tesi di Laurea

FUSIONE DI DATI STEREO E TIME-OF-FLIGHT MEDIANTE TECNICHE DI DEEP LEARNING

ANNO ACCADEMICO 2018-2019

25 SETTEMBRE 2019

Relatore:

Prof. PIETRO ZANUTTIGH

Laureando:

FRANCESCO PHAM

Correlatore:

Ing. GIANLUCA AGRESTI

Ai miei genitori,

che mi hanno sempre sostenuto.

Indice

Capitolo 1

Introduzione

La stima della profondità ha da sempre rappresentato un problema di notevole interesse. L'informazione sulla profondità è importante, ed in alcuni casi essenziale, per molteplici applicazioni pratiche della visione artificiale: guida autonoma, robotica, ricostruzione 3D e realtà aumentata sono soltanto alcune.

L'acquisizione delle geometrie tridimensionali di scene del mondo reale rappresenta da sempre un problema complesso, e gli strumenti sono stati accessibili soltanto in grosse compagnie e centri di ricerca. Nel corso degli anni, molte tecniche sono state sviluppate e nuovi dispositivi, dai costi più ridotti, sono stati introdotti nel mercato.

Le principali tecnologie per la percezione dell'ambiente sfruttano sensori ottici che catturano la luce visibile o infrarossa che viene riflessa dalla scena. Questo lavoro di tesi tratta due categorie di sensori in particolare, entrambi molto diffusi e accessibili: i sensori a visione stereoscopica e i sensori basati sul tempo di volo (Time-Of-Flight, ToF).

Nello stesso periodo, nel mondo della computer vision, il machine learning (e in particolare il deep learning) si è dimostrato uno strumento che ha permesso di ottenere risultati impensabili con altri metodi deterministici, in molti

problemi ritenuti complessi, fino a superare le prestazioni umane. E proprio grazie al machine learning è stato possibile sviluppare sistemi efficaci per la ricostruzione delle informazioni catturate da diverse tipologie di sensori.

1.1 Descrizione del progetto

L'obiettivo del lavoro di questa tesi è lo sviluppo di un sistema di machine learning per la fusione dei dati tridimensionali forniti dai sensori, cercando di fornire una ricostruzione più accurata. La scelta dei sensori gioca perciò un ruolo fondamentale: le telecamere stereo e il sensore ToF tendono ad avere caratteristiche complementari, e la loro combinazione si è rivelata efficace negli ultimi anni.

1.2 Principi di funzionamento del sistema di acquisizione

Il sistema di acquisizione è composto da una coppia di telecamere stereo e da un sensore Time-of-Flight disposti vicino. I due dispositivi catturano la stessa scena allo stesso istante.

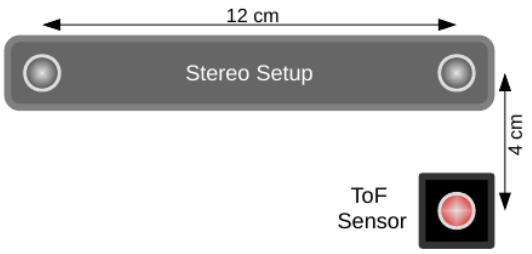


Figura 1.1: Rappresentazione del sistema di acquisizione ToF-Stereo. Il sensore ToF è posizionato sotto la telecamera di riferimento della coppia stereo [1].

1.2.1 Sistema di visione stereo

Il sensore a visione stereoscopica consiste nell'acquisire due immagini bidimensionali da una coppia di telecamere allineate che inquadrano la stessa scena. Successivamente, in fase di post-processing realizzato via software, viene effettuata una operazione di calibrazione detta rettificazione con lo scopo di rendere complanari le due immagini e allineate sullo stesso asse [13]. Lo stesso punto P dello spazio viene proiettato nel piano dell'immagine di ciascuna delle telecamere. I punti risultanti p_L e p_R, detti *omologhi*, hanno le stesse coordinate verticali, considerata la rettificazione dei sensori. Viene chiamata *disparità* lo scostamento tra le coordinate orizzontali:

$$d = u_L - u_R$$

Tramite questo valore è possibile determinare la posizione del punto P nello spazio.

Uno dei principali vantaggi è il costo ridotto delle telecamere CCD o CMOS, facilmente accessibili nel mercato. Tale sensore è inoltre passivo, quindi può sfruttare l'illuminazione dell'ambiente. Ciò permette di essere utilizzato in ambienti esterni, al contrario di altri sensori che sfruttano pattern proiettati con illuminatori. Inoltre può avere alte risoluzioni e una limitata quantità di rumore.

Lo svantaggio maggiore è la dipendenza dei risultati forniti da questi sensori dalla tessitura delle immagini utilizzata nel calcolo degli omologhi, ad esempio sono evidenti delle difficoltà nell'analisi di scene con pattern uniformi o ripetitivi. Pertanto essi presentano una robustezza solitamente limitata.

Figura 1.2: Sensore stereo ZED

1.2.2 Sensore Time-Of-Flight

Il sensore ToF determina le informazioni riguardo la profondità sulla base del fatto che le onde elettromagnetiche viaggiano alla velocità $c \approx 3 \times 10^8 [m/s]$. Il sensore emette un impulso luminoso (tipicamente infrarosso) il quale colpisce la scena e viene riflesso indietro. Viene dunque misurato il tempo τ che occorre al segnale luminoso per percorrere un tragitto pari al doppio della distanza dell'oggetto dalla telecamera ρ. La relazione che lega ρ e τ è

$$\rho = \frac{c\tau}{2}$$

Per misurare il tempo cercato e migliorare la risoluzione del sensore, l'impulso emesso è modulato secondo un segnale sinusoidale, così che anche l'eco abbia lo stesso andamento, ma sfasato e attenuato in ampiezza.

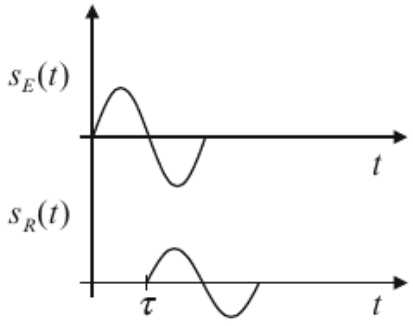


Figura 1.3: Segnale sinusoidale trasmesso e corrispondente segnale ricevuto da un sensore ToF [13]

Dispositivi ToF costituiti da un singolo trasmettitore e un singolo ricevitore, vengono tipicamente utilizzati in telemetri laser. Al fine di realizzare una mappa bidimensionale dell'ambiente, un sistema è quello di montare il sensore su una piattaforma roteante, tale configurazione ha trovato l'utilizzo in rilievi urbani, in architettura e recentemente anche nel campo dell'automobilistica. Il sistema utilizzato in questo lavoro di tesi appartiene bensì alla famiglia dei sensori ToF matriciali chiamati anche ToF camera [13].

ToF camera Nelle ToF camera, gli impulsi luminosi sono segnali infrarossi inviati tramite LED che illuminano l'intera scena mentre il ricevitore è una matrice di sensori CCD/CMOS. Diversamente dal meccanismo di tipo scanner, le ToF camera catturano le geometrie della scena in un singolo scatto nella quale ogni pixel misura, indipendentemente dalle altre, la distanza del punto della scena di fronte.

Figura 1.4: Il Microsoft Kinect è un esempio di ToF Camera

I vantaggi maggiori sono, al contrario delle telecamere stereo, l'indipendenza dal contenuto della scena, e la maggiore rapidità poiché non richiede algoritmi particolari per il calcolo della disparità.
Tuttavia, la tecnologia non è esente da alcune limitazioni:

- Per superfici poco riflettenti o di colore scuro, il segnale di ritorno è debole, ciò risulta in misure poco accurate.

- La limitata risoluzione spaziale non permette di ricavare una informazione dettagliata della geometria della scena.

- Il "multipath error" è un ulteriore problema che appare tipicamente vicino alle zone di incidenza tra le superfici. Esso è provocato dalla riflessione multipla del segnale luminoso prima di raggiungere il sensore [13].

1.3 Il Machine Learning

Il machine learning, o apprendimento automatico, è una branca dell'informatica che fornisce ai computer la capacità di imparare ad eseguire un task da un certo insieme di dati di training, senza essere esplicitamente programmati a farlo. In sostanza, il machine learning esplora l'utilizzo di metodi matematico-computazionali per apprendere informazioni direttamente dai dati, senza modelli matematici ed equazioni predeterminate. Gli algoritmi di apprendimento automatico migliorano le loro prestazioni in modo "adattivo" mano a mano che gli "esempi" da cui apprendere aumentano.

I problemi che il machine learning punta a risolvere vengono classificati in tre categorie:

- **Apprendimento supervisionato**: ogni istanza del set di esempi (training set) presenta gli input e i corrispondenti valori attesi in output. Questi esempi vengono presentati uno per volta alla macchina, il quale deve essere in grado di approssimare l'esatta natura della relazione presente tra l'input e il corrispondente output. L'obiettivo finale è dunque quello di insegnare al modello la capacità di predire correttamente i valori attesi su un set di istanze non presenti nel training set (test set). Questo scenario è quello trattato in questo lavoro di tesi.

- **Apprendimento non supervisionato**: al contrario dell'apprendimento supervisionato, gli output corrispondenti agli input forniti non sono conosciuti. All'algoritmo vengono presentati bensì numerosi dati dai quali la macchina deve estrarre le caratteristiche o i pattern nascosti.

Esistono svariati approcci per la progettazione di sistemi di machine learning che vanno dagli alberi di decisione agli algoritmi genetici. La tecnica di interesse per il lavoro di questa tesi è quello basato sulle reti neurali artificiali.

1.3.1 Reti neurali artificiali

Le reti neurali artificiali (in inglese *artificial neural network*, ANN) sono una classe di modelli matematici oggetto di numerosi studi. La loro importanza è derivata dal loro largo utilizzo in differenti ambiti e applicazioni. Da come suggerisce la parte "neurale" del nome, essi sono modelli che ricalcano il complesso funzionamento del sistema nervoso biologico. Le ANN sono composte da unità, chiamati neuroni, e le connessioni tra essi imitano il ruolo delle sinapsi.

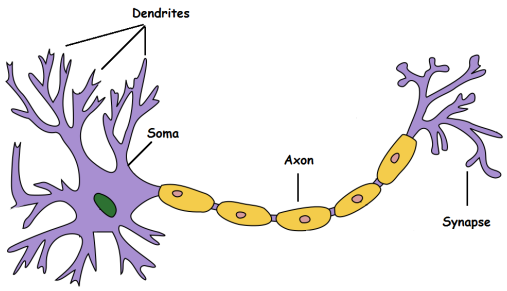


Figura 1.5: Neurone biologico [5]

Percettrone Il più semplice modello dei neuroni artificiali è il *percettrone*. Esso prende in input un vettore di valori reali e ne effettua una combinazione

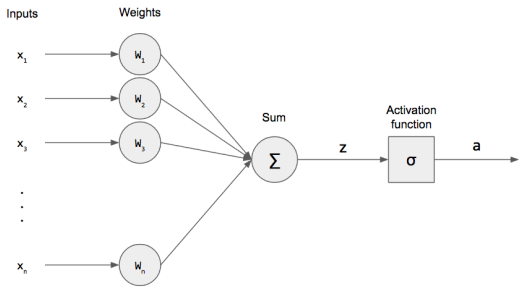


Figura 1.6: Schema di un percettrone [3]

lineare. Il risultato viene infine passato ad una funzione, detta funzione di attivazione, che restituisce l'output del neurone. Il modello complessivo può essere descritto dalla seguente formula:

$$y = \sigma(\sum_i w_i x_i + b)$$

dove w_i sono i pesi e b è un termine, detto bias, che viene aggiunto in seguito. Nel percettrone classico la funzione di attivazione σ è una funzione a gradino, tuttavia sono numerose le alternative: tra i più comuni sono il sigmoide e il ReLU.

Deep Neural Network Le reti neurali artificiali sono tipicamente organizzati in più strati, i nodi che la compongono sono suddivisi in tre macrocategorie. Abbiamo i nodi appartenenti allo strato di ingresso (input), i nodi appartenenti allo strato di uscita (output) e infine i nodi degli strati nascosti (hidden). L'architettura per strati è tipica delle reti neurali feed-forward. Tali reti sono caratterizzate dal fatto che l'attivazione dei nodi d'ingresso si propaga in avanti verso quelli dello strato (o degli strati) nascosto e da questi verso quelli dello strato di uscita.

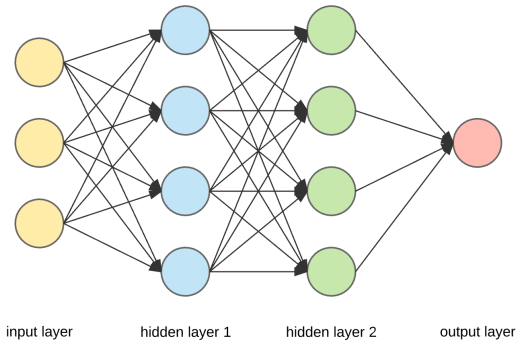


Figura 1.7: Rappresentazione di una rete neurale profonda [2]

1.3.2 Reti neurali convoluzionali

Le reti neurali visti fino ad ora sono composti da neuroni completamente interconnessi con ogni neurone del precedente e successivo strato. Ciò comporta un numero sproporzionato di pesi che devono essere gestiti all'aumentare degli strati e dei neuroni. Il problema diventa significativo dal punto di vista della memoria e del tempo di addestramento. Un altro svantaggio è l'*overfitting*. Infatti, avere un grosso numero di parametri, implica che se il training set non è abbastanza grosso, la rete potrebbe imparare a discriminare perfettamente gli esempi del training set, però avere scarse prestazioni su dati nuovi.

Questi problemi sono stati affrontati efficacemente con le reti neurali convoluzionali (*convolutional neural network*, CNN). Differentemente dalle tradizionali reti neurali, qui i neuroni sono connessi soltanto ad un sottoinsieme dei neuroni dello strato precedente, questa regione viene definita campo recettivo (*receptive field*). Tale modello si ispira alla struttura della corteccia visiva del mondo animale. Nelle CNN approssimiamo la risposta di un singolo neurone agli stimoli del proprio campo recettivo attraverso una operazione di convoluzione.

Convoluzione La convoluzione viene effettuata tra l'input di uno strato e uno o più filtri, detti anche *kernel*. L'output della operazione di convoluzione è detta *feature map*. Ci sono tante feature map quanti sono il numero di filtri utilizzati per elaborare gli input.

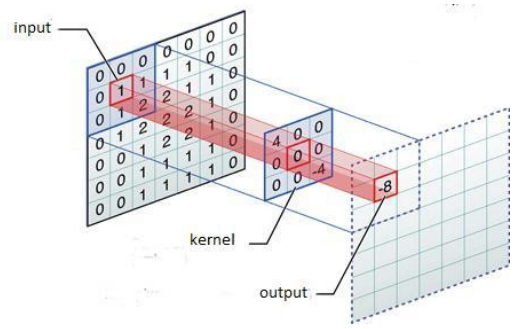


Figura 1.8: illustrazione della operazione di convoluzione con un filtro [4]

Il principio di convoluzione è fondamentale: oltre a mantenere limitato il numero di parametri, esso fornisce una proprietà di invarianza alla traslazione. Di fatto, lo stesso filtro viene applicato su tutta l'immagine, ciò permette di rilevare una particolarità nell'immagine ovunque essa sia, indipendentemente dalla sua posizione.

Funzioni di attivazione Analogamente alle reti neurali tradizionali, anche nelle CNN vengono usate le funzioni di attivazione. Questi vengono solitamente messi immediatamente dopo gli strati di convoluzione e hanno lo scopo di permettere alla rete l'apprendimento di funzioni non lineari. La funzione di attivazione più utilizzata in questo campo negli ultimi anni è il rettificatore lineare (*Rectified Linear Unit*, ReLU). Questa, rispetto ad altre funzioni come la tangente iperbolica o la sigmoide, pone rimedio al problema del *vanishing gradient*. Nelle reti molto profonde, infatti, le derivate sempre più in profondità possono diventare molto piccole, il che rende il training difficile. Essendo il

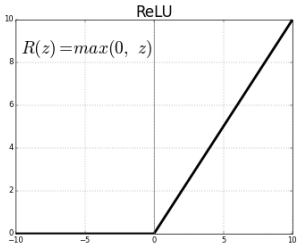


Figura 1.9: funzione di attivazione ReLU [10]

gradiente della ReLU pari a 1 per i valori positivi e applicando la regola di derivazione a catena non si incorre in valori tendenti a zero. Ciò porta ad un processo di allenamento tendenzialmente più rapido.

Capitolo 2

Acquisizione dei dati

Come già accennato, l'addestramento delle reti neurali convoluzionali richiede un dataset. Il training set deve essere sufficientemente grande in modo che la rete possa generalizzare in modo appropriato, cioè dare risultati plausibili per input che non ha mai visto. Al momento, tuttavia, l'acquisizione accurata di dati ground truth per scene 3D reali non è una operazione immediata. Per questa ragione, in questo lavoro verrà utilizzato un dataset sintetico simulato con *Blender* composto da 55 scene 3D differenti. Le scene contengono diverse problematiche di acquisizione che includono riflessi, diversi tipi di illuminazione, e pattern ripetitivi.

2.1 Preparazione dei dati

Per poter compiere una corretta fusione delle informazioni tridimensionali del ToF camera e del sensore stereo, bisogna fare in modo che i dati forniti siano all'interno dello stesso sistema di riferimento.

Interpolazione e riproiezione Il sensore Time-of-Flight fornisce una mappa di profondità della scena. Per poter eseguire la fusione con i dati restituiti dal sensore stereo, bisogna tenere conto le diverse risoluzioni. Infatti, il sensore ToF ha una risoluzione nettamente inferiore rispetto alle moderne telecamere a colori, perciò è necessario interpolare le immagini restituite in modo da portarlo alla risoluzione delle telecamere stereo.
Inoltre, le informazioni sulla profondità del sensore ToF vengono riproiettate sulla prospettiva della telecamera a colori di riferimento che, senza perdita di generalità, assumiamo essere la telecamera destra della coppia stereo.

Calcolo della disparità Il sensore Time-of-Flight e il sensore stereo rappresentano la scena 3D in modo differente. Come già visto, il sistema stereo ritorna una mappa di disparità dove ogni pixel rappresenta la differenza di posizione tra i punti omologhi nelle due immagini. La ToF camera d'altro canto ritorna una mappa di profondità nella quale ogni pixel esprime la distanza tra il sensore e il corrispondente punto sulla scena. La disparità e la profondità sono legate da una relazione di proporzionalità inversa:

$$z = \frac{b \cdot f}{d}$$

dove z è la distanza, d è la disparità, b è la baseline dello stereo e f è la lunghezza focale delle telecamere [13]. La disparità è stata scelta come unità di misura comune, dunque la mappa di profondità del ToF viene convertita in una mappa di disparità in modo tale da rendere omogenee le rappresentazioni dei due sensori.

Data augmentation Per *data augmentation* si intende una tecnica che consiste nell'apportare alcune modifiche casuali nelle istanze del dataset, come rotazioni, ritagli, traslazioni e altre operazioni di image processing. Lo scopo

è quello di ampliare il numero di esempi nel training set per una maggiore robustezza e variabilità, e di conseguenza ridurre le probabilità di overfitting. Per questo lavoro sono stati effettuati casualmente operazioni di ritaglio, ribaltazione orizzontale e verticale dell'immagine e rotazione di $\pm 5°$. Il prodotto di queste operazioni è un training set composto da 3902 esempi di dimensioni 128x128.

Capitolo 3

Fusione tramite deep learning

3.1 Le componenti della CNN

3.1.1 Reti neurali residuali

Le reti neurali residuali, o ResNet, sono state introdotte da Microsoft nel 2015 battendo il record della competizione ILSVRC con un errore del 3.6% [7], superando per la prima volta le prestazioni umane.
L'idea che sta alla base delle reti neurali residuali è l'utilizzo di un particolare tipo di blocco, il *residual block*, che sfrutta il concetto delle *skip connection*.
Vediamo ora nella figura 3.1 il modello del residual block:

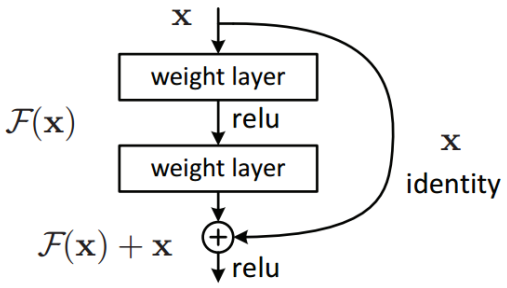


Figura 3.1: Schema del residual block [7]

Ogni blocco consiste in una serie di strati che rappresentano una certa $F(x)$, seguito da una *skip connection* che aggiunge al risultato l'input del blocco stesso. Dunque in uscita dal blocco si ha

$$H(x) = F(x) + x$$

Attraverso la concatenazione di blocchi di questo tipo, ResNet impara a predire l'output non attraverso l'apprendimento di una trasformazione diretta dei dati di input, ma attraverso l'apprendimento del termine $F(x)$, detto *residuo*, da sommare al dato di input per arrivare all'output. Tale modello semplifica la costruzione di strati di identità, basta infatti spingere $F(x)$ verso zero e lasciare l'output come x. Ciò permette agli strati più profondi della rete di imparare solo il contributo incrementale rispetto a quanto già appreso nei layer precedenti, preservando così le informazioni. Questa architettura ha permesso di creare reti molto più profonde, fino a 152 strati.

3.1.2 Batch Normalization

Durante il processo di addestramento, gli esempi del training set vengono processati in gruppi, detti mini-batch, di dimensione fissa per velocizzare la fase di train sfruttando il parallelismo offerto dalle GPU. La *Batch Normalization* è una tecnica che consiste nel normalizzare i dati del mini-batch ad ogni strato intermedio della rete. In questo modo si riduce il così detto *Internal Covariance Shift* [8]. L'utilizzo del batch normalization ha l'intento di velocizzare il training della rete neurale e migliorarne la stabilità e le performance. Questa operazione si è rivelata particolarmente efficace nelle reti neurali residuali.

3.1.3 Bottleneck

Il modello ResNet introduce il così detto design a "collo di bottiglia". L'idea che sta dietro all'utilizzo di questo design è quello di ridurre le dimensioni dell'input del blocco residuale prima di effettuare la convoluzione 3x3. La

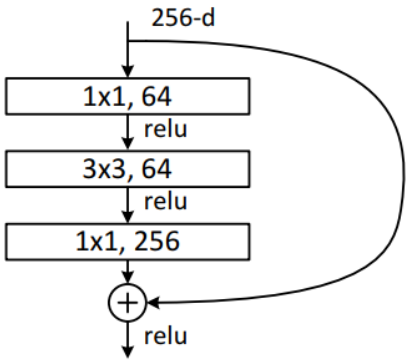


Figura 3.2: Il design a collo di bottiglia utilizzato dal ResNet [7]

riduzione della dimensionalità viene eseguita con uno strato di convoluzione 1x1, lasciando allo strato di convoluzione 3x3 delle dimensioni di input/output più ridotte, tenendo così un numero di parametri molto più ristretto. Infine, l'ultimo strato 1x1 ha lo scopo di ripristinare le dimensioni a quelle iniziali. Questo permette un tempo di addestramento più veloce, senza pesare sulle prestazioni [7].

3.1.4 Convoluzione dilatata

Il vantaggio della convoluzione dilatata è la possibilità di catturare più informazioni dall'input, espandendo il campo recettivo di una convoluzione 2D.
In realtà, anche utilizzando un filtro di dimensioni maggiori si ha un receptive field allargato, ad esempio una convoluzione 3x3 dilatata con rate $r = 2$ ha lo stesso campo recettivo di una normale convoluzione 5x5. Tuttavia, il numero

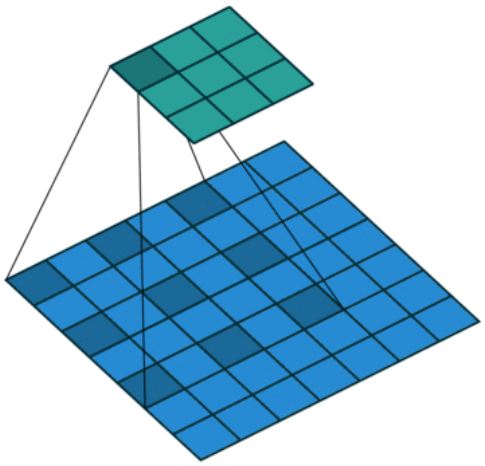


Figura 3.3: Illustrazione della convoluzione dilatata [11]

di parametri del modello con dilatazione può essere notevolmente ridotto. Un kernel $k \times k$ mantiene, infatti, $k \times k$ parametri avendo però un campo recettivo pari a $k_e \times k_e$ con $k_e = k + (k-1)(r-1)$ [12].

3.2 la rete neurale selezionata

La rete neurale convoluzionale utilizzata in questo lavoro di tesi, alla quale ci riferiremo con la sigla DRFNet (*dilated residual fusion network*), è formata da una rete neurale residuale con bottleneck, batch normalization e convoluzione dilatata. Nella figura 3.4 viene illustrato il blocco residuale utilizzato.
La DRFNet differisce dalla ResNet per l'assenza dello strato finale *fully connected*, si tratta quindi di una rete *fully convolutional* che ha come output una singola feature map con la stessa risoluzione delle immagini in input. L'architettura selezionata è una concatenazione di sei blocchi residuali, seguito da una convoluzione che combina le 128 feature map restituendo l'immagine risultante.

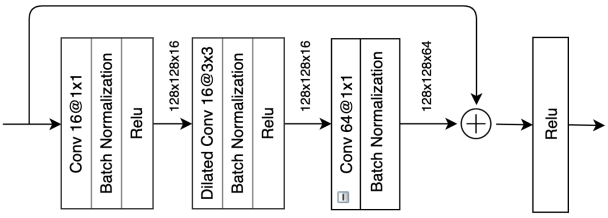


Figura 3.4: Diagramma del blocco residuale

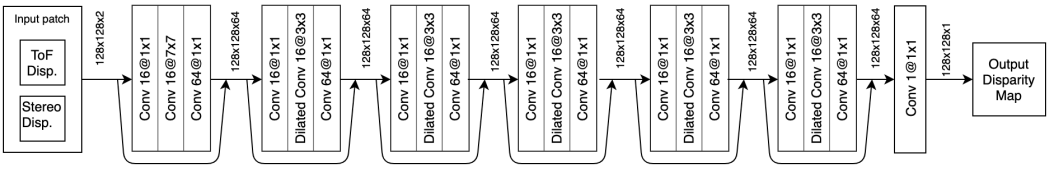


Figura 3.5: Diagramma della DRFNet. Per sintesi non ho incluso gli strati di normalizzazione e rettificazione lineare.

3.3 Il processo di training

3.3.1 Loss function e ottimizzazione

Loss function Per poter minimizzare l'errore di output della rete neurale è necessario definire tale errore.

Le due funzioni di loss valutate in questo lavoro sono le norme l1 e l2, detti anche *mean absolute error* e *mean squared error*.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y_j}|$$

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y_j})^2$$

Da un veloce confronto, le due norme danno risultati sperimentali simili. È stato deciso di utilizzare la norma l2, una delle loss function maggiormente utilizzate.

Ottimizzazione Il training è stato effettuato con algoritmo di ottimizzazione *Adam*. Questo metodo di ottimizzazione è una estensione della discesa stocastica del gradiente che ha visto un largo utilizzo in molte applicazioni del deep learning nella computer vision e nell'elaborazione del linguaggio naturale. Il nome *Adam* deriva da *adaptive moment estimation*. Questo metodo combina i vantaggi di AdaGrad e RMSProp, modificando i parametri di learning rate in modo adattivo in base alle stime dei momenti di primo e secondo ordine dei gradienti [9]. Il learning rate scelto è $lr = 0.001$ mentre i fattori di decadimento impostati sono $\beta_1 = 0.9$ e $\beta_2 = 0.999$

3.3.2 Inizializzazione delle variabili

L'inizializzazione dei pesi della rete neurale è una delicata operazione che se non effettuata bene potrebbe impedire la convergenza della rete durante l'apprendimento. Infatti, se i pesi vengono inizializzati troppo piccoli allora andando in profondità nella rete il segnale comincia ad attenuarsi, fino a diventare inutilizzabile. D'altra parte, se i pesi sono troppo grandi allora il segnale in ingresso si amplifica fino a saturare i neuroni. Impostare i pesi a zero ha poco senso in quanto renderebbe la rete equivalente a un modello lineare. Perciò la soluzione è inizializzare i pesi in modo casuale secondo una certa distribuzione adeguata.
La distribuzione utilizzata in questo lavoro è quello proposto da Xavier. L'idea è quella di utilizzare una distribuzione gaussiana con media nulla e una certa varianza finita. Consideriamo un neurone lineare:

$$y = w_1 x_1 + w_2 x_2 + ... + w_N x_N + b$$

Mano a mano che si processano i vari strati, vogliamo che la varianza rimanga la stessa. Questo aiuta ad evitare che il segnale esploda oppure che si attenui

verso lo zero. In altre parole vogliamo inizializzare i pesi in modo che la varianza rimanga uguale per x e y.
Calcoliamo ora la varianza di y:

$$Var(y) = Var(w_1x_1 + w_2x_2 + ... + w_Nx_N + b)$$

Assumiamo che le variabili x_i siano indipendenti e calcoliamo le varianze dei termini all'interno delle parentesi di destra. Abbiamo:

$$Var(w_ix_i) = E(x_i)^2Var(w_i) + E(w_i)^2Var(x_i) + Var(w_i)Var(x_i)$$

Abbiamo assunto che gli input e i pesi abbiano media nulla, perciò abbiamo:

$$Var(w_ix_i) = Var(w_i)Var(x_i)$$

Notiamo che b è costante perciò sparisce. Sostituiamo nell'equazione iniziale:

$$Var(y) = Var(w_1)Var(x_1) + ... + Var(w_N)Var(x_N)$$

Dato che sono identicamente distribuiti, possiamo scrivere:

$$Var(y) = N * Var(w_i) * Var(x_i)$$

Perciò, se vogliamo che la varianza di y sia uguale alla varianza di x, allora il termine $N * Var(w_i)$ deve valere 1. Dunque:

$$N * Var(wi) = 1$$

$$Var(w_i) = \frac{1}{N}$$

Abbiamo trovato la formula di **inizializzazione di Xavier**, non dobbiamo fare altro che inizializzare i pesi secondo una normale a media nulla e varianza pari all'inverso del numero di connessioni in ingresso [6].

Capitolo 4

Analisi dei risultati

4.1 Framework e ambiente di lavoro

Per la generazione del dataset a partire dai dati acquisiti è stato utilizzato l'ambiente MATLAB, nel quale sono state anche fatte le operazioni di data augmentation come ritaglio, ribaltazione e rotazione. Mentre per lo sviluppo del modello sono state utilizzate le API Keras su backend TensorFlow in linguaggio di programmazione Python.
L'ambiente sul quale è stato eseguito il training è quello fornito da Google Colaboratory in cloud ossia un sistema con Ubuntu 18.04 e scheda grafica Tesla K80.

4.2 Training e validation

Con l'architettura presentata e una volta pronto il codice necessario, è stato possibile procedere con il training della rete e con la successiva valutazione dei risultati. Gli iperparametri dell'addestramento sono stati impostati nel seguente modo:

- durata del training: 75 epoche
- learning rate: 0.001
- batch size: 32, il più alto possibile con la memoria GPU a disposizione.
- loss function: MSE

Il tempo per completare il training è stato di circa 1 ora e 40 minuti.
In figura 4.1 e 4.2 sono mostrati i grafici degli andamenti del loss di training e validation.

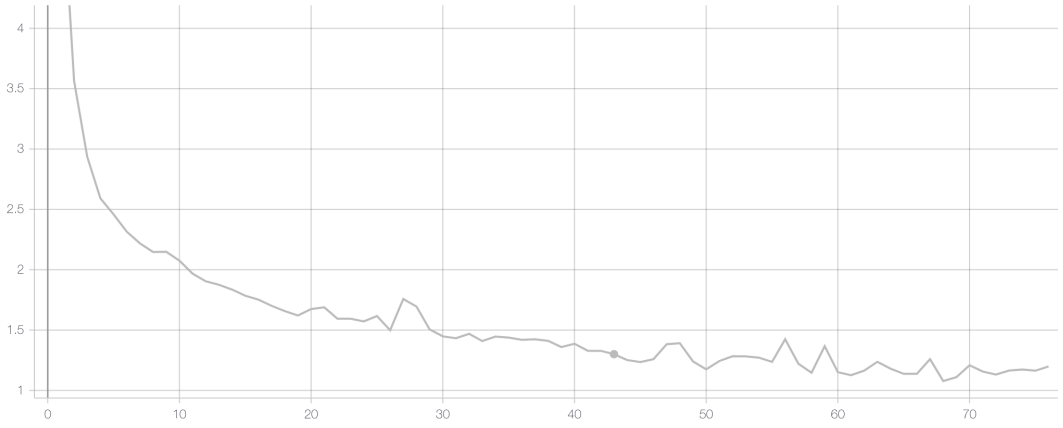


Figura 4.1: Andamento del training loss

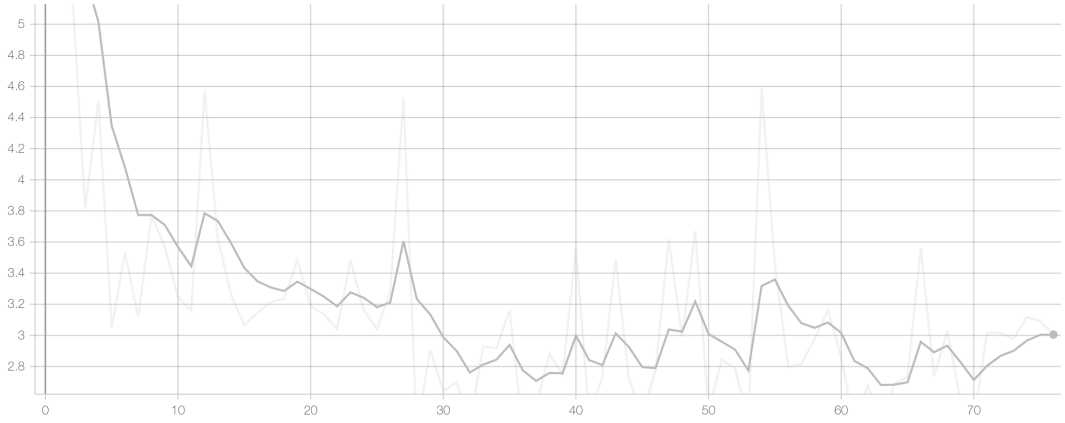
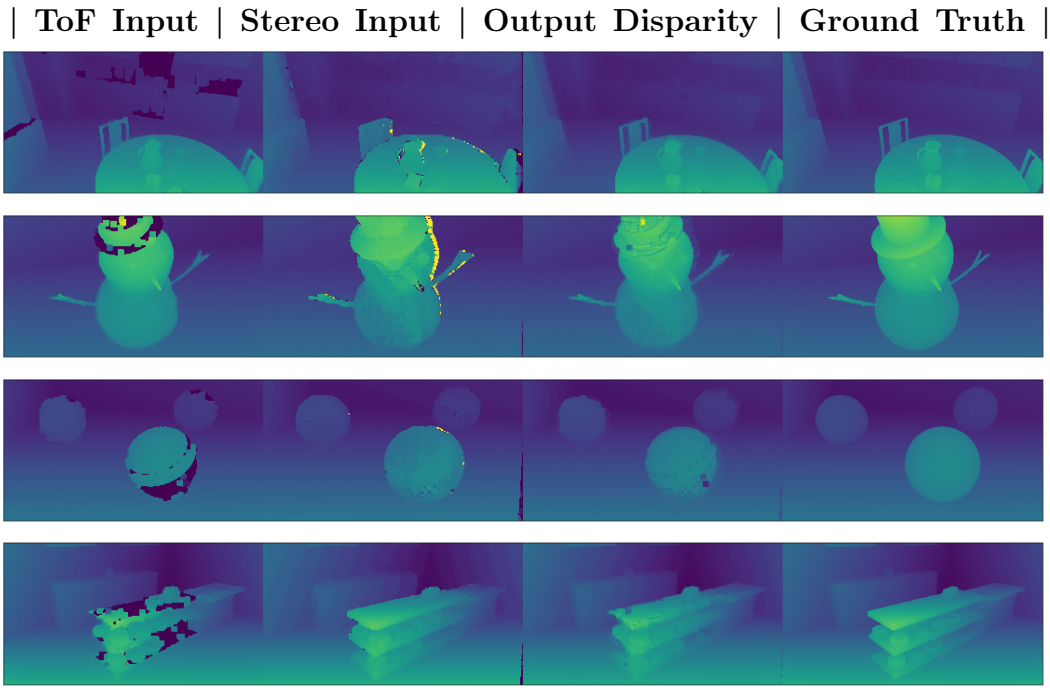


Figura 4.2: Andamento del validation loss

4.3 Test e risultati sperimentali

Nelle figure seguenti sono mostrati alcuni risultati sperimentali della rete addestrata. La prima e la seconda colonna mostrano gli input ToF e Stereo rispettivamente. Si nota nei dati ToF la presenza di alcuni "buchi" sulle superfici poco riflettenti e sulle regioni affette dal "multipath error", ossia in prossimità delle zone di incidenza tra le superfici. Per quanto riguarda lo stereo, si notano in particolare accuratezze limitate sui bordi.
La terza colonna mostra la fusione, ossia l'output della rete. Infine nella quarta colonna vengono illustrati i ground truth. La fusione è in grado di estrarre le informazioni più accurate da entrambe le sorgenti, tappando le lacune del ToF ed eliminando gli artefatti lungo i bordi dello stereo.



4.4 Confronti tra differenti architetture

La nostra rete contro una rete elementare Confrontiamo ora i risultati che abbiamo ottenuto contro quelli acquisiti da una semplice rete illustrata in figura 4.3. Questa architettura non include blocchi residuali, normalizzazioni e convoluzioni dilatate.

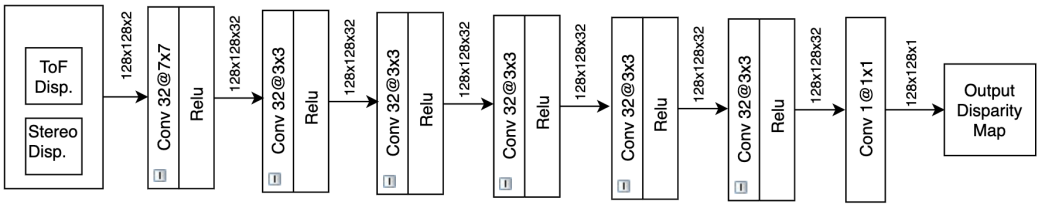


Figura 4.3: Diagramma di una semplice CNN senza blocchi residuali, convoluzioni dilatate e normalizzazioni

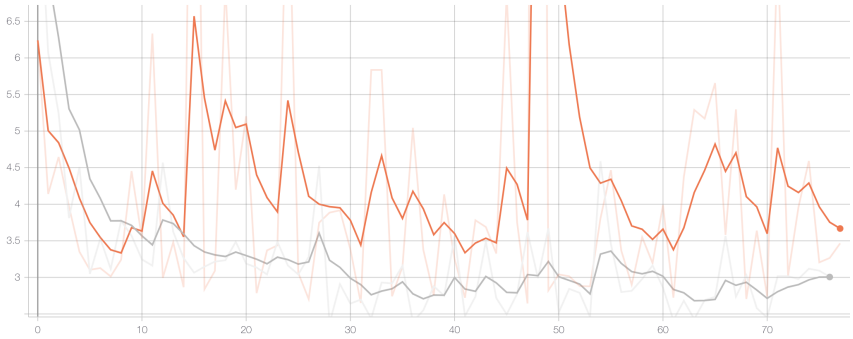


Figura 4.4: Confronto del validation loss di DRFNet contro una versione più elementare.

DRFNet (figura 3.5)

La CNN elementare (figura 4.3)

Si nota immediatamente dalla figura 4.4 che la nostra rete performa assai meglio della rete con architettura elementare convergendo molto più rapidamente e con meno oscillazioni.

Convoluzione dilatata contro senza dilatazione Proviamo ora a togliere la dilatazione dagli strati di convoluzione della nostra rete. Proviamo quindi a comparare i risultati così ottenuti con quelli acquisiti con la dilatazione.

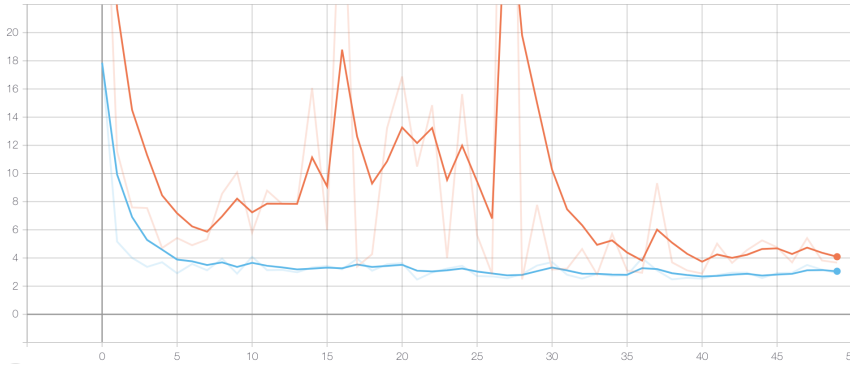


Figura 4.5: Confronto del validation loss della rete con strati di convoluzione dilatata contro senza dilatazione.

- DRFNet (figura 3.5)
- La rete senza dilatazione sulla convoluzione

Come si vede dalla figura 4.5, la dilatazione negli strati di convoluzione ha un effetto positivo sulle performance della rete sul validation set. È possibile difatti apprezzare un andamento migliore e più stabile nel caso dell'utilizzo della dilatazione.

Comparazioni tra diverse profondità della rete Con l'obiettivo di determinare la migliore profondità della rete, proviamo a variare il numero di blocchi residuali tra 4, 6 e 8.

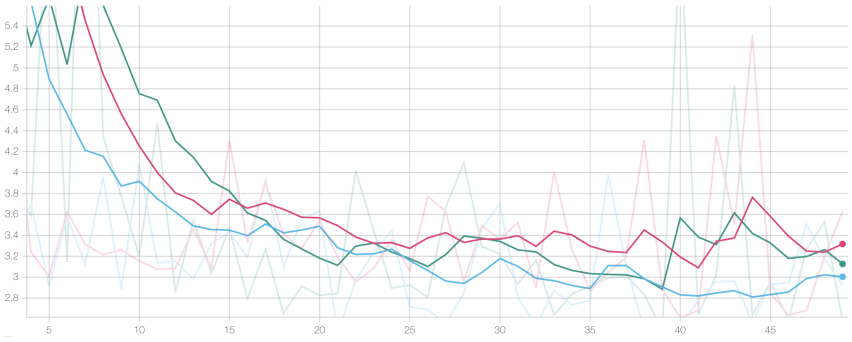


Figura 4.6: Confronto tra differenti numeri di blocchi residuali.

6 blocchi residuali

4 blocchi residuali

8 blocchi residuali

Dal grafico comparativo di figura 4.6 si vede un andamento leggermente migliore nel caso dell'utilizzo di 6 blocchi residuali, con 8 blocchi si hanno risultati molto simili mentre il caso peggiore è quello con 4 blocchi.

Capitolo 5

Conclusioni

L'obiettivo di questa tesi è stata quella di costruire un sistema in grado di fondere i dati tridimensionali forniti da due diversi tipi di sensori, ossia un sensore a tempo di volo e una coppia di telecamere stereo.

I risultati riportati hanno mostrato come l'utilizzo di tecniche di deep learning all'avanguardia, permetta di realizzare un modello in grado di sfruttare al meglio le informazioni fornite dai due sensori, realizzando una ricostruzione più accurata delle strutture tridimensionali della scena catturata. In particolare si è visto come l'utilizzo delle reti neurali residuali assieme alle convoluzioni dilatate abbia effettivamente apportato benefici tangibili sulle performance della rete nella stima della disparità.

I risultati ottenuti da questo lavoro di tesi possono inoltre essere applicati per la fusione di una diversa combinazione di sensori al fine di ottenere risultati simili oppure migliori.

Possibili sviluppi futuri comprendono sicuramente il test della rete neurale addestrata, su un dataset reale per verificare le performance del modello su dati del mondo reale.

Bibliografia

[1] Gianluca Agresti, Ludovico Minto, Giulio Marin, and Pietro Zanuttigh. Stereo and tof data fusion by learning from synthetic data. 2018.

[2] Arden Dertat. Applied deep learning - part 1: Artificial neural networks. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6, 2017.

[3] Mohit Deshpande. Perceptrons: The first neural networks. https://pythonmachinelearning.pro/perceptrons-the-first-neural-networks/, 2017.

[4] Alejandro Escontrela. Convolutional neural networks from the ground up. https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1, 2018.

[5] Maël Fabien. The rosenblatt's perceptron. https://maelfabien.github.io/deeplearning/Perceptron/#why-neurons.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

[8] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[10] Kanchan Sarkar. Relu : Not a differentiable function: Why used in gradient based optimization? and other generalizations of relu. `https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7` 2018.

[11] Kanchan Sarkar. Review: Dilatednet — dilated convolution (semantic segmentation). `https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5`, 2018.

[12] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2016.

[13] Pietro Zanuttigh, Giulio Marin, Carlo Dal Mutto, Fabio Dominio, Ludovico Minto, and Guido Maria Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras*. Springer, 2016.