Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

Dipartimento di Informatica - Scienza e Ingegneria DISI

Laurea Magistrale in Ingegneria Informatica

TESI DI LAUREA

in

Sistemi Digitali M

Percezione depth guidata mediante proiezione virtuale di pattern

Candidato:
Luca Bartolomei

Relatore:
Prof. Stefano Mattoccia

Correlatori:
Dott. Matteo Poggi
Dott. Fabio Tosi

Sessione II
Anno Accademico 2021/2022

Indice

Introduzione

L'utilizzo di informazioni di profondità (*depth*) è oggi di fondamentale utilità per molteplici settori applicativi come la robotica, la guida autonoma o assistita, la realtà aumentata e il monitoraggio ambientale. Questa tipologia di informazione permette di interagire con l'ambiente direttamente o indirettamente [1, 2].

Per inferire distanze rispetto agli oggetti inquadrati nella scena, in commercio si posso acquistare sensori depth di due tipologie distinte: attivi e passivi. I sensori attivi effettuano misurazioni su segnali emessi dai sensori stessi, per esempio le tecnologie come le camere *Time of Flight (ToF)* (e.g., Microsoft Kinect V2), a luce strutturata (e.g., Microsoft Kinect) e i sistemi *Light Detection And Ranging (LiDAR)* a scansione multi linea (e.g., Velodyne). Tuttavia ogni tipologia di sensore attivo presenta diverse limitazioni tra le quali la risoluzione limitata, il costo, le dimensioni ingombranti (e.g., LiDAR), l'incapacità di funzionare in alcuni ambienti (e.g., i sensori cwToF non funzionano bene nelle scene *outdoor*) o in presenza di alcuni materiali (e.g., superfici assorbenti) [1].
I sensori passivi invece ricavano le informazioni di profondità dall'ambiente senza emettere segnali, bensì utilizzando i segnali provenienti dall'ambiente (e.g., luce solare).

Possiamo scindere ulteriormente la tipologia dei sensori passivi in sistemi stereo e sistemi monoculari. I sistemi stereo sfruttano due camere poste ad una distanza prefissata (*baseline*) per calcolare la profondità, utilizzando un modello geometrico che si inspira alla visione binoculare degli umani (biomimetica). Invece i sistemi monoculari fanno affidamento ad una sola camera e tipicamente ad una rete neurale artificiale per effettuare la stima di profondità. Rispetto al sistema stereo si ha un potenziale mercato più ampio (smartphone di fascia media e bassa), tuttavia hanno performance peggiori dato che si tratta di una stima relativa: possiamo stimare la profondità relativa tra gli oggetti presenti nella scena mediante delle assunzioni geometriche della scena [1, 2, 3, 4].

Nei sensori depth passivi stereo è richiesto un algoritmo per elaborare le immagini delle due camere: la tecnica di *stereo matching* viene utilizzata appunto per stimare la pro-

fondità di una scena, producendo in uscita una mappa di disparità (*disparity map*), che in seguito potrà essere trasformata in una mappa di profondità (*depth map*), sfruttando il modello geometrico. Lo stato dell'arte utilizza tecniche di *deep learning* raggiungendo buoni risultati, tuttavia queste tecniche richiedono una grande quantità di dati strutturati per funzionare: la raccolta dei dati, in particolare delle mappe di disparità, non può essere effettuata manualmente per ragioni pratiche. Occorre dunque utilizzare degli automatismi per raccogliere nuovi dati o per raffinare dati già esistenti [2, 3, 4, 5].

Di recente la ricerca si è occupata anche della sinergia con sensori attivi al fine di migliorare la stima della depth ottenuta da un sensore stereo: si utilizzano i punti affidabili generati dal sensore attivo per guidare l'algoritmo di stereo matching verso la soluzione corretta. Risulta quindi fondamentale sfruttare al massimo i punti affidabili cercando di minimizzarne il numero, dato il costo elevato di quest'ultimi. In letteratura è stata affrontata la tematica in varie maniere: riducendo lo spazio di ricerca, ottimizzando i costi o introducendo una penalità [6, 7].

In questa tesi si è deciso di affrontare questa tematica da un punto di vista nuovo, utilizzando un sistema di proiezione virtuale di punti corrispondenti in immagini stereo: i *pixel* delle immagini vengono alterati per guidare l'algoritmo ottimizzando i costi. I pixel sono alterati in base ai punti affidabili ottenuti precedentemente, però, a differenza di [7], l'ottimizzazione viene applicata all'inizio della *pipeline* di matching: si tratta di una metodologia che è completamente esterna all'algoritmo di matching utilizzato, precisando anche che le due tecniche non sono mutualmente esclusive.

Un altro vantaggio della strategia proposta è la possibilità di iterare il processo, andando a cambiare il pattern in ogni passo: aggregando i passi in un unico risultato, è possibile migliorare il risultato finale in modo simile al metodo Spacetime stereo [8]. Tuttavia quest'ultimo utilizza un proiettore reale sulla scena e effettua un'aggregazione nel dominio del tempo, invece che nel dominio delle iterazioni. Integrare nel tempo limita l'applicazione alle sole scene statiche e l'utilizzo di un proiettore limita l'applicazione a scene *indoor*.

Si possono ottenere i punti affidabili sia mediante sensori attivi (e.g., LiDAR o ToF), sia direttamente dalle immagini, stimando la confidenza delle mappe prodotte dal medesimo sistema stereo: la confidenza permette di classificare la bontà di un punto nella mappa di disparità. Nel corso della tesi sono stati utilizzati sensori attivi per verificare l'efficacia della proiezione virtuale. Infine sono state effettuate analisi sulle misure di confidenza: lo scopo è verificare se le misure di confidenza possono rimpiazzare o assistere i sensori attivi.

1. Percezione della profondità

In questo capitolo analizzeremo i sensori che permettono di percepire la profondità, tra cui i sensori passivi, dove viene applicata la strategia innovativa proposta. L'obiettivo di un sensore di profondità è di catturare le informazioni di depth di un oggetto o una scena per permetterne la rappresentazione e l'elaborazione 3D. Per ogni tipologia di sensore saranno introdotti i concetti fondamentali che la contraddistinguono.

1.1 Rappresentazione dei dati

Tutti i sensori di profondità dovranno restituire dati sulla scena osservata: vi è quindi la necessità di una rappresentazione dei dati 3D. Nel caso di sensori depth, che processano immagini (e.g., sensori passivi stereo), è necessario fornire al sensore i dati dell'immagine con una certa rappresentazione 2D.

1.1.1 Rappresentazione 2D

Le immagini digitali rappresentano le informazioni luminose che arrivano al piano d'immagine attraverso il centro ottico della camera. I dati sono raccolti in una matrice bidimensionale, dove ogni cella viene chiamata pixel: esso rappresenta l'intensità luminosa registrata in una determinata zona nel piano d'immagine[1].

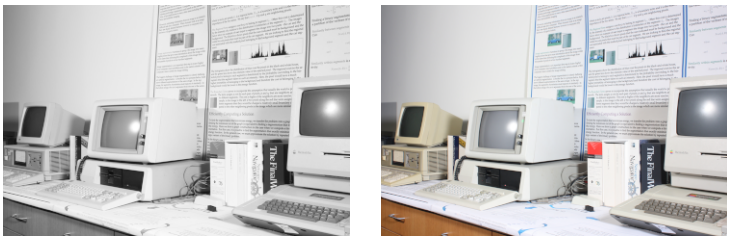


Immagine a scala di grigi Immagine a colori

Figura 1.1: esempio di immagine in due rappresentazioni differenti[2].

[1]Il modello è descritto in dettaglio nella sezione 2.1.

Immagini a scala di grigi

Se i pixel dell'immagine contengono solamente un'intensità luminosa, allora si parlerà di immagine a scala di grigi o grayscale: l'intensità luminosa della scala è codificata all'interno di un solo canale per ogni pixel. Le immagini a scala di grigi sono più leggere sia da immagazzinare sia da elaborare, dato che devono contenere solo l'informazione di luminosità e non l'informazione cromatica. Ciò può essere un vantaggio nelle situazioni in cui non è strettamente necessario conoscere l'informazione cromatica (e.g., analisi dei bordi di un'immagine).

Immagini a colori

Per rappresentare immagini a colori è necessario utilizzare uno spazio di colore (e.g., sRGB, AdobeRGB): permette di riprodurre un insieme di colori (*gamut*) su più dispositivi, utilizzando un modello di colore e una funzione di mappatura [9].

Il modello di colore (e.g., RGB, CMYK) descrive il colore matematicamente, mentre la funzione di mappatura associa i colori reali (i.e., provenienti da uno spazio di colore assoluto, come il CIE 1931) al colore del modello: un valore RGB di $(123, 53, 210)$ sarà mappato in due colori reali distinti rispettivamente per i due spazi di colore sRGB e AdobeRGB.

RGB è il modello più comunemente utilizzato ed è di tipo additivo: la miscelazione delle componenti primarie dà luogo al colore bianco. I colori primari del modello sono rosso (Red), verde (Green) e blu (Blue), per cui necessita di 3 canali per essere codificato: elaborare un'immagine a colori è 3 volte più costoso rispetto ad un'immagine a scala di grigi. Per questo motivo gli algoritmi di matching tradizionali[3] utilizzano immagini a scala di grigi.

1.1.2 Rappresentazione 3D

Le rappresentazioni dei dati in 3D vengono utilizzate per la memorizzazione, l'elaborazione e la riproduzione di superfici di oggetti in 3D: il volume di un oggetto sarà determinato dalla sua superficie chiusa.

I dati tridimensionali possono essere raggruppati in due categorie: strutturati e non strutturati.

[2]Fonte immagine: Middlebury 2014 - Vintage [5].

[3]Vedi sezione 2.3

- I dati non strutturati (e.g., formato XYZRGB) si limitano a elencare i dati ed eventuali connessioni senza una topologia, risulta quindi spesso necessario delle elaborazioni aggiuntive per estrarre informazioni di più alto livello (e.g., piani).
- I dati strutturati hanno una qualche struttura a supporto (e.g., informazioni semantiche).

Tra le tipologie di rappresentazione 3D disponibili, ci limiteremo alle sole utilizzate nell'elaborato.

Point cloud

Si tratta di una rappresentazione 3D molto semplice: vengono rappresentati punti 3D sotto forma di tupla (x, y, z). Alcuni formati (e.g., PLY, XYZRGB) codificano anche i canali RGB, espandendo la tupla in (x, y, z, r, g, b). I punti sono relativi ad un sistema di riferimento: le point cloud prodotte nel testo utilizzano il SRF[4].

Vista dall'alto Vista frontale

Figura 1.2: esempio di point cloud colorato in due viste generato dalla Fig. 1.1, usando la strategia proposta.

Depth map

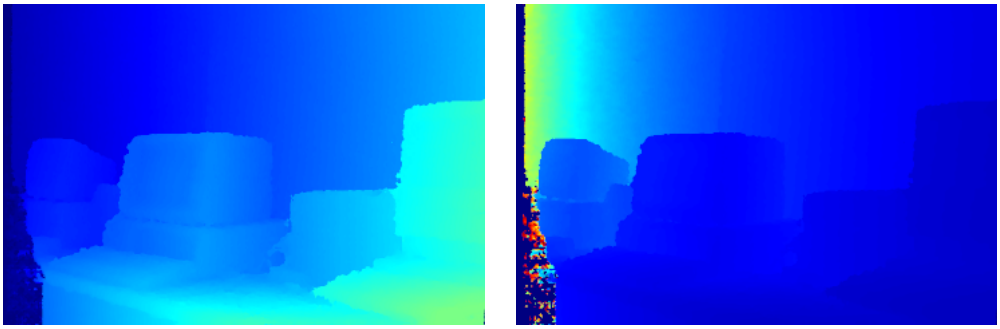
La mappa di profondità (depth map) è un'immagine a scala di grigi dove ogni pixel codifica la distanza rispetto ad un riferimento (i.e., CRF). Tuttavia, per una migliore rappresentazione visiva, si utilizzano trasformazioni (*colormap*) per colorare una depth map (Fig. 1.3): ad ogni valore di intensità è associato un colore. Dato che i canali

[4]Vedi la sottosezione 2.1.1.

delle immagini occupano generalmente 8 bit, le distanze vanno codificate (e quantizzate) all'interno del range intero $0 \dots 255$.

Figura 1.3: mappatura dei valori interi $0 \dots 255$ su colorazione "Jet": i colori più freddi (blu) rappresentano valori prossimi allo zero (e.g., distanze ravvicinate o valori di disparità bassi), mentre i colori più caldi rappresentano valori alti.

Questa rappresentazione è la più comune tra i sensori depth attivi, dato che è compatta quanto un'immagine. Tuttavia richiede dei parametri aggiuntivi di calibrazione per la ricostruzione 3D.
I sensori depth passivi stereo invece utilizzano una mappa di disparità (disparity map): la disparità rappresenta la differenza in pixel della posizione degli oggetti osservati da due punti di vista. La disparity map è trasformabile mediante i parametri del sensore in una mappa depth[5].



Mappa di disparità Mappa di profondità

Figura 1.4: disparity map e depth map generate dalla Fig. 1.1, usando la strategia proposta[6].

L'utilizzo di questi dati aiuta l'estrazione di informazioni di alto livello (*features*) (e.g., segmentazione semantica, SLAM, analisi BLOB).

[5] In dettaglio nella sezione 2.2
[6] Mappe basate sulla scena Vintage - Middlebury 2014 [5].

1.2 Sensori attivi

Come accennato nell'introduzione, in commercio si possono acquistare sensori di tue tipologie distinte: attivi e passivi. I sensori attivi emettono un segnale fisico (e.g., luminoso, acustico) per il calcolo della distanza. Analizzeremo il principio ToF alla base delle tecnologie LiDAR e camere cwToF e introdurremo anche il principio a luce strutturata.

1.2.1 Principio ToF

Il principio *Time of Flight (ToF)* è basato sull'intervallo di tempo (Δt) che impiega un impulso luminoso dall'istante in cui viene emesso e fino all'istante in cui torna alla sorgente di emissione, dopo essere riflesso dall'ambiente [10]. L'impulso effettua un'andata e un ritorno: l'intervallo di tempo misurato risulterà approssimabile a due volte la distanza tra il sensore e l'oggetto (*Round Trip Time (RTT)*), trascurando il rumore.

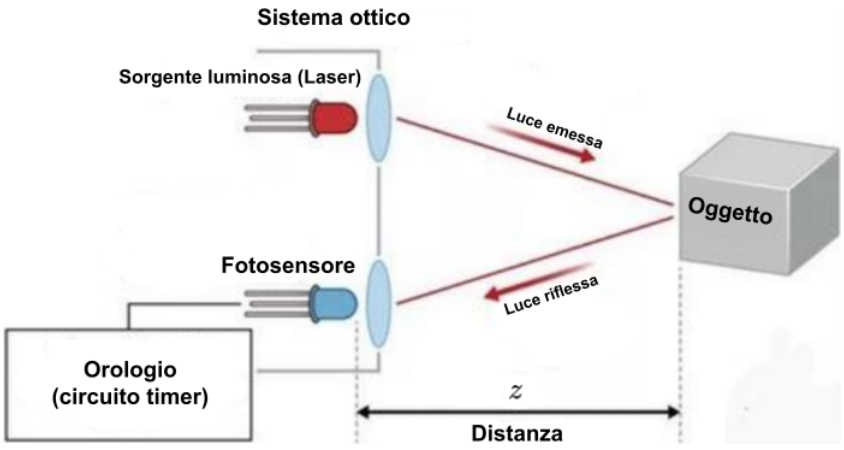


Figura 1.5: rappresentazione del principio *Time of Flight (ToF)*: si possono notare il sistema ottico e il timer[7].

La depth (z) è ricavata moltiplicando la velocità della luce (c) con l'intervallo di tempo (Δt):

$$2z = \int_0^{\Delta t} c\, dt = c\Delta t \tag{1.1}$$

Utilizzando un clock digitale per misurare il tempo, l'intervallo (Δt) sarà discretizzato con un intervallo di tempo minimo $\Delta t_0 = \frac{1}{f}$, con (f) la frequenza dell'orologio digitale, di

[7]Fonte immagine: [10] (tradotta).

conseguenza la distanza minima misurabile è $2l_0 = c\Delta t_0 = \frac{c}{f}$. Manipolando l'equazione 1.1, otteniamo la distanza (z) in base al conteggio (n) eseguito dall'orologio digitale:

$$z = \sum_{i=1}^{n} \frac{c}{2f} = n\frac{c}{2f} = nl_0 \tag{1.2}$$

Limitazioni

I fattori che influenzano l'accuratezza delle tecnologie ToF riguardano il sistema ottico (i.e., *edge* dell'impulso, *bandwith* del ricevitore, *Signal to Noise Ratio (SNR)*) e l'accuratezza dell'orologio digitale. Il sistema ottico richiede un impulso laser potente e molto corto, con tempi di accensione e spegnimento ripidi (*edge raising* e *edge falling*), mentre per tolleranze basse sono richiesti orologi digitali con un Δt_0 nell'ordine dei picosecondi. In particolare il SNR è dovuto a fenomeni non ideali, come *photon shot noise*, rumore nel circuito e interferenza *multi-path* (*scattering*).

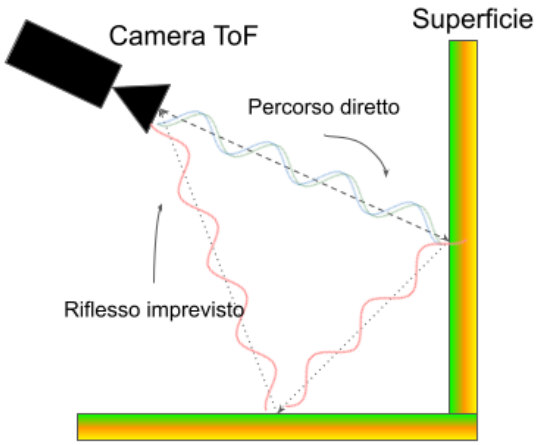


Figura 1.6: rumore dovuto ad interferenza multi-path.

1.2.2 LiDAR

Le tecnologie LiDAR utilizzano il principio ToF e un impulso laser: sono utilizzate in applicazioni come la guida autonoma per raccogliere informazioni sull'ambiente e sulla vettura (e.g., *Simultaneous Localization And Mapping (SLAM)*), mentre in altri contesti non vengono utilizzate, per via degli svantaggi di ingombro e costo. Possiamo dividere i LiDAR in due categorie: a scansione e senza scansione.

Figura 1.7: esempi di sensori LiDAR a scansione e senza scansione[8].

LiDAR a scansione

I LiDAR a scansione sono caratterizzati da un motore che fa ruotare il dispositivo, oltre che il sistema ottico e l'orologio digitale. Possono essere divisi in altre due categorie: a singola linea e multi linea: entrambe le categorie hanno una certa risoluzione angolare, ciò implica che la linea ottenuta scansionando l'ambiente non è continua. I primi effettuano una scansione su un solo piano, mentre i secondi combinano più piani permettendo di ottenere una point cloud più dettagliata a discapito di ingombro e costo.

LiDAR senza scansione (3D Flash Lidar)

Questa tipologia di LiDAR non possiede alcuna parte meccanica: il laser non illuminerà un singolo punto, bensì un'area, necessitando però di un ricevitore più complesso per mantenere una buona accuratezza. Ogni pixel del ricevitore verrà processato individualmente: viene misurato l'RTT dell'impulso per calcolare la distanza secondo il principio ToF.

1.2.3 Camere cwToF

Il sistema di misurazione del tempo utilizzato nelle tecnologie LiDAR è molto costoso, dato che è necessario misurare intervalli dell'ordine del picosecondo per raggiungere tolleranze del millimetro. Per ridurre i costi legati alla misurazione del tempo si utilizza il principio continuous wave Time of Flight (cwToF) basato sulla differenza di fase tra segnale ricevuto e segnale inviato [11]: il segnale è trasmesso continuamente ed è modulato secondo una portante (e.g., sinusoidale, quadra) ad una certa frequenza f_m.

[8]Fonte immagini: [10].

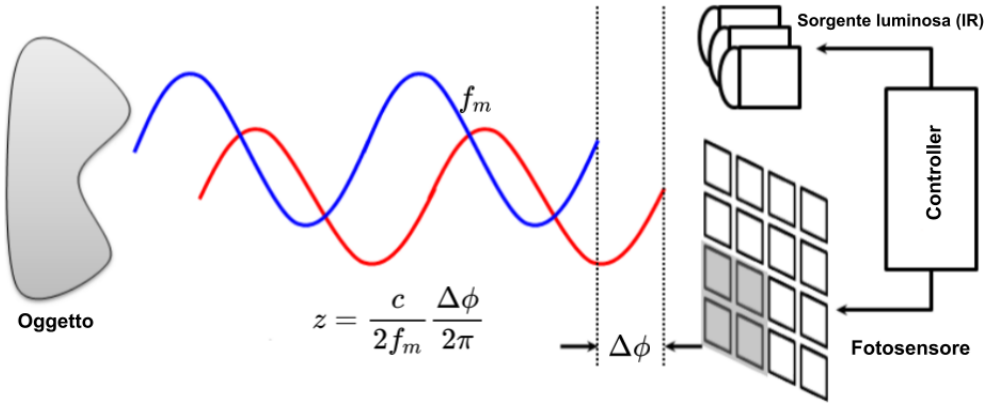


Figura 1.8: principio cwToF: si può notare lo scostamento di fase $\Delta\phi$ tra segnale emesso e ricevuto[9].

Dopo aver ottenuto lo scostamento di fase ($\Delta\phi$), possiamo ricavare l'intervallo di tempo (Δt) considerando la fase ϕ descritta in termini di velocità angolare ($2\pi f_m$):

$$\Delta\phi = 2\pi f_m \Delta t \rightarrow \Delta t = \frac{\Delta\phi}{2\pi f_m} \tag{1.3}$$

Una volta ricavato l'intervallo di tempo (Δt), possiamo andarlo a sostituire nella formula 1.1:

$$z = \frac{c}{2f_m}\frac{\Delta\phi}{2\pi} \tag{1.4}$$

Per ricavare lo scostamento di fase si utilizza la cross-correlazione tra il segnale ricevuto e il segnale inviato (e.g., metodo *4-bucket phase*).

Questo adattamento permette di utilizzare frequenze molto più basse, rendendo l'orologio digitale più semplice e meno costoso: di conseguenza anche il sistema ottico può essere semplificato, sostituendo il laser con diodi IR. La sorgente luminosa a infrarosso rende limitata l'applicazione outdoor: la luce solare è fonte di disturbo per questa tipologia di ricevitori.

Tuttavia l'introduzione dello scostamento di fase porta ad un'ambiguità intrinseca (*phase wrapping*): posso misurare distanze senza ambiguità solo all'interno del range $[0, \frac{c}{2f_m}]$. Il problema può essere risolto (*phase unwrapping*) diminuendo la frequenza f_m per aumentare il range a discapito dell'accuratezza, oppure utilizzando più portanti a frequenze diverse per distinguere la vera distanza.

[9]Fonte immagine: [11] (tradotta).

1.2.4 Luce strutturata

I sensori depth a luce strutturata utilizzano un pattern noto proiettato sulla scena tramite un proiettore e una camera per rilevare come il pattern ha avvolto gli oggetti: la differenza tra il pattern distorto e quello noto permette di generare una mappa di disparità e di calcolare la profondità con grande accuratezza [12].

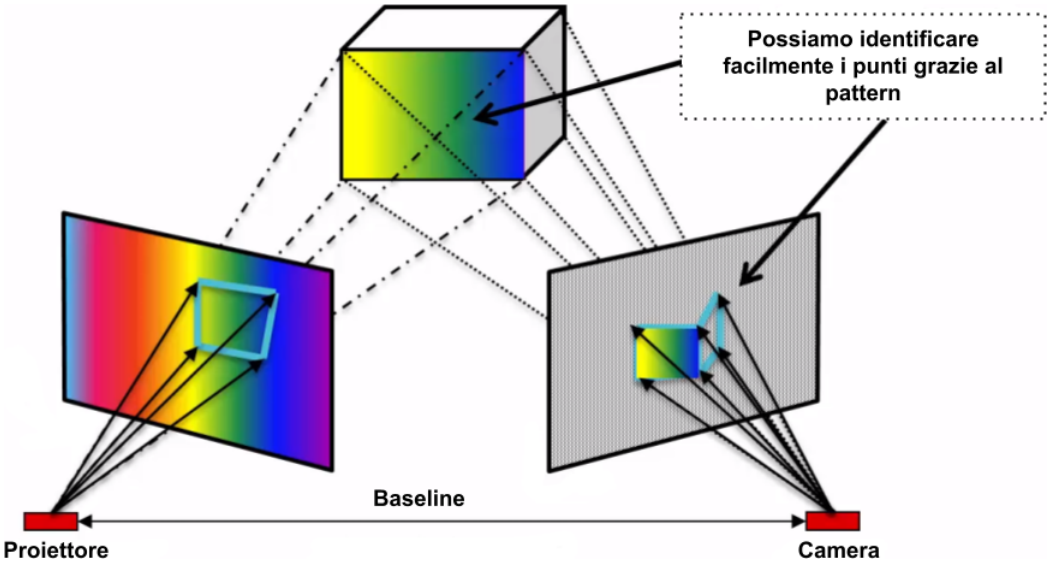


Figura 1.9: principio a luce strutturata[10]: a differenza del sistema stereo, una camera è sostituita dal proiettore.

L'utilizzo del proiettore permette di risolvere l'ambiguità presente nei sistemi stereo[11]: il pattern è progettato in modo da essere altamente distinguibile, risolvendo a priori il problema della corrispondenza tramite un *mapping* da numeri (*codewords*) a insiemi di pixel.

Possiamo catalogare i codewords in tre categorie: basati sul tempo (*time-multiplexing*), basati sullo spazio (*spatial neighbourhood*) e diretti (*direct codification*).

Anche se l'accuratezza di questa tecnologia è elevata, non è possibile l'utilizzo in ambienti outdoor: fonti luminose esterne (e.g., sole, altri proiettori) interferiscono con il processo di acquisizione. Inoltre, in presenza di alcuni materiali (e.g., vetro), il sensore a luce strutturata può fallire nel rilevare la profondità.

[10]Fonte immagine (tradotta):
`https://training.ti.com/introduction-how-structured-light-works`

[11]Vedi sottosezione 2.2.1: problema della corrispondenza.

1.3 Sensori passivi

I sensori di profondità passivi permettono di ricavare la profondità con i soli segnali presenti nell'ambiente: la scena deve fornire abbastanza informazioni per poter stimare la profondità e quindi non è efficace in situazioni di scarsa visibilità o su certe tipologie di superfici. I sensori passivi vengono caratterizzati per il numero di punti di vista con cui si osserva la scena (i.e., monoculare, stereo, *multi-view*) e su come questi punti di osservazione vengono catturati (e.g., da camere, in sequenza da una sola camera) [2, 13]. Nel testo approfondiremo i sensori stereo[12], perché utilizzati nella strategia proposta[13].

1.3.1 Sensore stereo

I sensori stereo e i sensori multi-view permettono di determinare la profondità mediante triangolazione, la quale è possibile grazie all'osservazione della scena da due (o più) prospettive diverse. I sensori multi-view generalizzano il caso del sensore stereo, in quanto la scena è osservata da un numero arbitrario di punti di vista e non solo due.

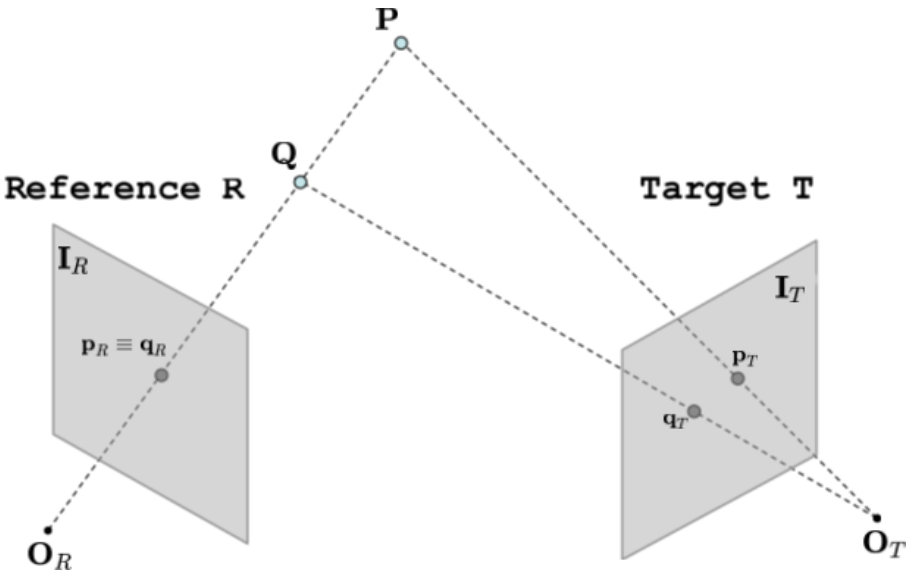


Figura 1.10: modello stereo generico[14].

Il sistema stereo è caratterizzato da due camere poste ad una certa distanza l'una dall'altra (baseline): generalmente questa configurazione non viene utilizzata, bensì si allineano le camere sullo stesso piano orizzontale per formare il sistema stereo "laterale"[15].

[12] In dettaglio nel Capitolo 2.
[13] Descritta nel Capitolo 3.
[14] Fonte immagine: [2] (adattata).
[15] In dettaglio alla sezione 2.2

Figura 1.11: sensore stereo laterale: le due camere sono poste sullo stesso asse orizzontale[16].

Il sistema stereo laterale ha dei vincoli geometrici molto stretti: ciò implica che è praticamente impossibile allineare le due camere meccanicamente. Per risolvere il problema, si ricorre alla tecnica della rettificazione[17], la quale permette di allineare virtualmente le camere.

La triangolazione dei punti 3D avviene conoscendo la baseline e la posizione dei punti sui piani d'immagine delle camere (problema della corrispondenza[18]). Dato che il problema è intrinsecamente ambiguo (posso a priori associare un punto nell'altra vista), si aggiungono una serie di vincoli per rendere la ricerca più facile. Questa tipologia di sensori funzionano bene sia in ambienti al chiuso sia all'aperto, tuttavia è necessario un algoritmo (stereo matching) per risolvere il problema della corrispondenza: la strategia proposta[19] mira ad aiutare gli algoritmi esistenti, utilizzando punti 3D affidabili.

1.3.2 Sensore monoculare

I sensori monoculari cercano di stimare la profondità da una sola vista, tuttavia si tratta di un problema intrinsecamente ambiguo: esistono infinite posizioni valide per un punto 3D osservato dalla vista. Per affrontare il problema sono state ideate tecniche che fanno uso di algoritmi tradizionali (e.g., *shape-from-shading*, *shape-from-silhouette*) e altre che fanno uso di deep learning (e.g., MonoDepth2, PyD-Net) [1, 13, 14]. Gli algoritmi tradizionali non hanno riscosso molto successo: solo con l'avvento delle tecniche di deep learning i sensori monoculari hanno visto un miglioramento.

[16] Fonte immagine: [13].
[17] Vedi sottosezione 2.2.2.
[18] Per approfondimenti vedi sottosezione 2.2.1.
[19] Vedi Capitolo 3.

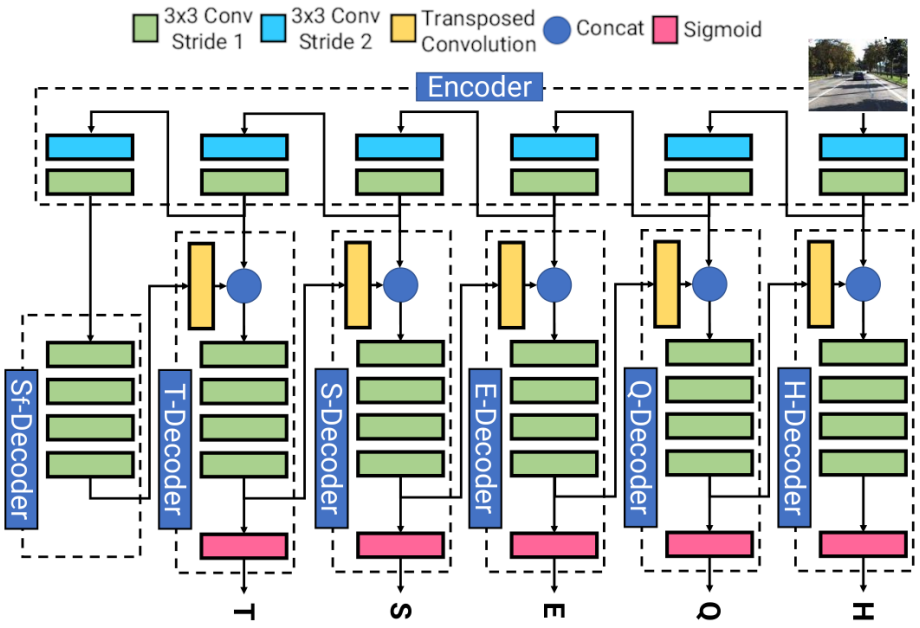


Figura 1.12: architettura di PyD-Net: data la sua struttura piramidale (*coarse-to-fine*), è molto leggera e può essere eseguita su molti *device embedded*.[20]

Tuttavia l'utilizzo di deep learning implica una strategia di addestramento (e.g., *supervised*, *semi-supervised*, *self-supervised*) robusta e un dataset molto ampio e variegato per assicurare una robustezza in ambienti eterogenei. Nella letteratura sono state proposte soluzioni come il mescolamento di dataset [4] e l'utilizzo di strategie di addestramento *teacher-student* [1].

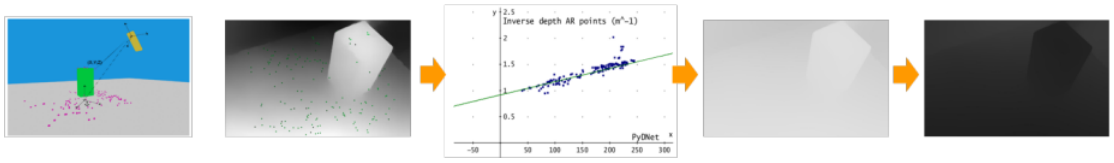


Figura 1.13: strategia di trasformazione delle distanze basata su punti 3D sparsi assoluti e RANSAC.[21]

Un altro problema dei sensori monoculari è che non forniscono una profondità rispetto ad un sistema di riferimento assoluto (e.g., CRF, SRF, WRF), bensì solo una distanza relativa tra oggetti più vicini e oggetti più lontani: è necessario trasformare le distanze relative in distanze assolute (Fig. 1.13) [1].

[20] Fonte immagine: [14].
[21] Fonte immagine: [1].

2. Stereo matching

Nel seguente capitolo tratteremo in dettaglio il sensore passivo stereo, partendo dal modello della camera, come si utilizza il modello per formare un sistema stereo, a cosa serve l'algoritmo di stereo matching in un sistema stereo ed infine lo stato dell'arte. Questa trattazione ci aiuterà a comprendere le fondamenta su cui si basa la strategia proposta.

2.1 Modello geometrico pinhole

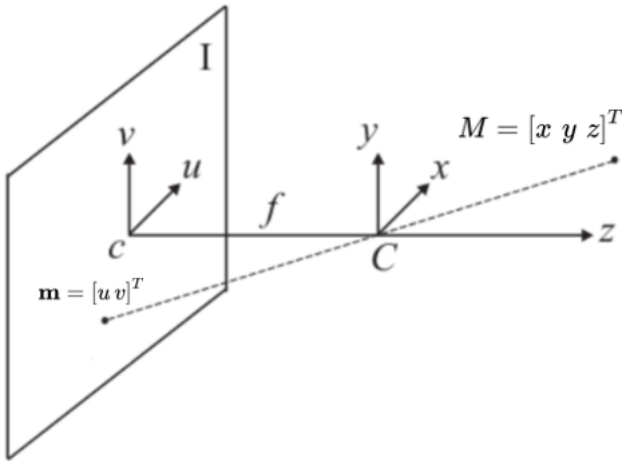


Figura 2.1: modello geometrico pinhole[1].

Il modello (Fig. 2.1) viene utilizzato per modellare l'acquisizione delle immagini in una camera, cioè su come viene sfruttata la luce riflessa dalla scena 3D e incidente sul piano d'immagine (**I**) per comporre una rappresentazione 2D della scena stessa. Il modello si basa su un piccolo foro (*pinhole*), da cui passano solo i raggi luminosi rettilinei che partono dall'oggetto della scena e raggiungono il piano d'immagine tramite esso.

[1]Fonte immagine: [15] (modificata).

Oltre che il piano d'immagine **I**, dove i raggi colpiscono la sua superficie per formare un'immagine, troviamo:

- il centro ottico (**C**) cioè il pinhole;
- il centro sul piano d'immagine (**c**) attraversato dall'asse del centro ottico;
- la lunghezza focale (f) cioè la distanza tra il piano d'immagine e il centro ottico;

Quest'ultima variabile è utilizzata nelle trasformazioni non-lineari, per passare dalla rappresentazione 3D ($\mathbf{M} = [x\ y\ z]^T$) alla rappresentazione 2D ($\mathbf{m} = [u\ v]^T$), espresse nello spazio euclideo $\mathbb{R}^3$ (consideriamo come unità di misura i millimetri [mm]) [15].

$$\frac{u}{x} = \frac{v}{y} = -\frac{f}{z} \quad \rightarrow \quad u = -\frac{f}{z}x, \quad v = -\frac{f}{z}y \tag{2.1}$$

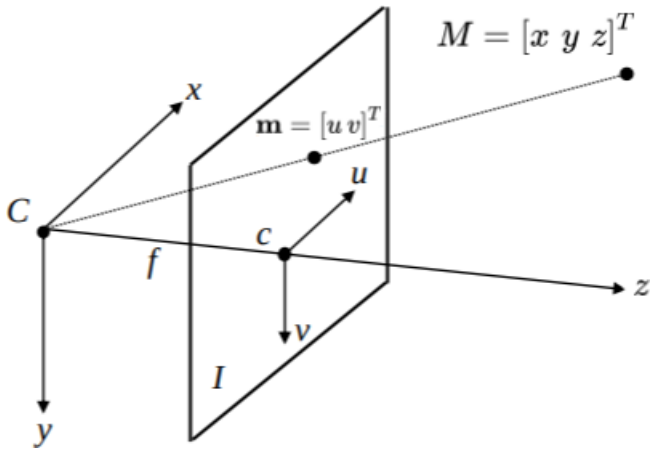


Figura 2.2: modello geometrico pinhole con piano d'immagine davanti[2].

Per eliminare il segno negativo dalla equazione 2.1, possiamo immaginare il piano d'immagine **davanti** al centro ottico (Fig. 2.2), ottenendo:

$$\frac{u}{x} = \frac{v}{y} = \frac{f}{z} \quad \rightarrow \quad u = \frac{f}{z}x, \quad v = \frac{f}{z}y \tag{2.2}$$

[2]Fonte immagine: [15] (modificata).

2.1.1 Coordinate omogenee

Tuttavia non è conveniente utilizzare la trasformazione 2.2 nello spazio euclideo $\mathbb{R}^3$, dato che si tratta di una equazione non lineare: le trasformazioni lineari possono essere rappresentate mediante matrici, molto utili nel calcolo parallelo.
Possiamo estendere lo spazio euclideo utilizzando le coordinate omogenee, ottenendo lo spazio prospettico $\mathbb{P}^3$: rispetto alle coordinate euclidee (x, y, z) si aggiunge una coordinata k, ottenendo $(kx, ky, kz, k), \forall k \neq 0$.

- Per $k \neq 0$ rappresentiamo coordinate euclidee, a meno di un fattore di scala k, inoltre possiamo ritornare alle coordinate euclidee standard dividendo ogni coordinata per k.
- Per $k = 0$, rappresentiamo punti all'infinito, non previsti nelle coordinate euclidee, ma che possono apparire nel piano d'immagine [15, 16].

Utilizzando le coordinate omogenee per rappresentare il modello pinhole, otteniamo la seguente forma matriciale:

$$k\tilde{\mathbf{m}} = \begin{bmatrix} ku \\ kv \\ k \end{bmatrix} = k \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = k \begin{bmatrix} \frac{f}{z}x \\ \frac{f}{z}y \\ 1 \end{bmatrix} = k \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \tilde{\mathbf{P}}\tilde{\mathbf{M}} \tag{2.3}$$

Definiamo $\tilde{\mathbf{m}}$ il vettore del punto sull'immagine in $\mathbb{P}^2$, con $\tilde{\mathbf{M}}$ il vettore del punto sulla scena in $\mathbb{P}^3$ secondo il riferimento CRF, infine con $\tilde{\mathbf{P}}$ la matrice di proiezione prospettica (PPM).

Per rendere il modello più realistico occorre inoltre tenere in conto della digitalizzazione delle immagini (parametri intrinseci), della rototraslazione (parametri estrinseci) del sistema di coordinate con origine nel centro ottico (*Camera Reference Frame (CRF)*), rispetto all'origine di un sistema di coordinate spaziali esterno (*World Reference Frame (WRF)*), della non perpendicolarità tra l'asse del centro ottico e il piano d'immagine (*skew*) e delle distorsioni delle lenti [16].
In particolare la rototraslazione verrà accennata solo per il sistema stereo laterale, in quanto verrà utilizzato il CRF della camera di sinistra come riferimento globale (*Stereo Reference Frame (SRF)*): tutte le rappresentazioni 3D presenti (e.g., Fig. 1.2) saranno con sistema di coordinate SRF. Nemmeno lo skew verrà considerato, perché solitamente trascurabile nei moderni sensori a stato solido [3].

2.1.2 Parametri intrinseci

Vengono modellati due aspetti della digitalizzazione: il campionamento e la traslazione del centro d'immagine (Fig. 2.3).

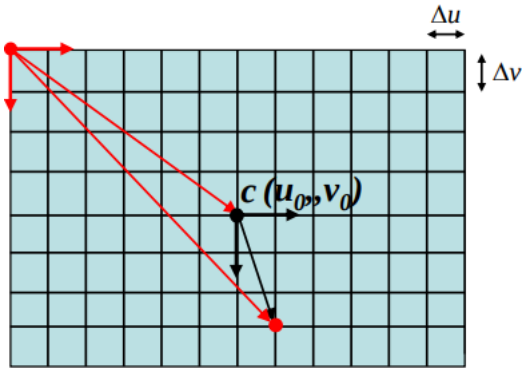


Figura 2.3: digitalizzazione dell'immagine[3].

Il campionamento riguarda la trasformazione di un piano d'immagine continuo in una matrice bidimensionale $M \times N$, dove ogni elemento è chiamato pixel: per tenere in conto del fenomeno, si introducono due fattori di scala aggiuntivi nelle trasformazioni non lineari:

$$[\text{mm}]\ u = \frac{f}{z}x \quad \rightarrow \quad [\text{px}]\ u = \frac{1}{\Delta u}\frac{f}{z}x = k_u\frac{f}{z}x, \quad k_u = \frac{1}{\Delta u}\ \left[\frac{\text{px}}{\text{mm}}\right] \tag{2.4}$$

$$[\text{mm}]\ v = \frac{f}{z}y \quad \rightarrow \quad [\text{px}]\ v = \frac{1}{\Delta v}\frac{f}{z}y = k_v\frac{f}{z}y, \quad k_v = \frac{1}{\Delta v}\ \left[\frac{\text{px}}{\text{mm}}\right] \tag{2.5}$$

Mentre la traslazione del centro d'immagine viene modellata rispetto all'origine del sistema di coordinate dell'immagine digitale (Fig. 2.3, in alto a sinistra):

$$u = k_u\frac{f}{z}x \quad \rightarrow \quad u = k_u\frac{f}{z}x + u_0 \tag{2.6}$$

$$v = k_v\frac{f}{z}y \quad \rightarrow \quad v = k_v\frac{f}{z}y + v_0 \tag{2.7}$$

[3]Fonte immagine: [16].

Ottenendo infine la trasformazione lineare in coordinate omogenee, apportando modifiche alla matrice di proiezione prospettica $\tilde{\mathbf{P}}$:

$$\tilde{\mathbf{P}} = \begin{bmatrix} k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} k_u f & 0 & u_0 \\ 0 & k_v f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathbf{A}\big[\mathbf{I} \,|\, \mathbf{0}\big] \tag{2.8}$$

Dove la matrice $\big[\mathbf{I} \,|\, \mathbf{0}\big]$ rappresenta la matrice di proiezione prospettica canonica (i.e., x/z, y/z), mentre la matrice $\mathbf{A}$ rappresenta i parametri intrinseci della camera modellata. Possiamo riscrivere $\alpha_u = k_u f$ e $\alpha_v = k_v f$, ottenendo la lunghezza focale in pixel e riducendo il numero delle incognite a quattro.

2.1.3 Parametri estrinseci

Solitamente il riferimento per il sistema di coordinate si trova al di fuori della camera (WRF). Bisogna quindi collegare questo riferimento al CRF: il legame tra i due sistemi di riferimento è modellato con una matrice di rotazione $\underset{3\times3}{\mathbf{R}}$ e un vettore di traslazione $\underset{3\times1}{\mathbf{T}}$:

$$\mathbf{W} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \rightarrow \quad \mathbf{M} = \mathbf{R}\mathbf{W} + \mathbf{T} \tag{2.9}$$

Possiamo riscrivere la formula 2.9 usando le coordinate omogenee:

$$\tilde{\mathbf{W}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{M}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \rightarrow \quad \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{W}} = \mathbf{G}\tilde{\mathbf{W}} \tag{2.10}$$

La matrice $\mathbf{G}$ (matrice dei parametri estrinseci) codifica la posizione e l'orientamento della camera (CRF) rispetto al sistema di riferimento esterno (WRF).
Possiamo riscrivere la matrice di proiezione prospettica (PPM) $\tilde{\mathbf{P}}$, tenendo conto sia dei parametri intrinseci sia dei parametri estrinseci, in una forma concisa:

$$\tilde{\mathbf{P}} = \mathbf{A}\big[\mathbf{R} \,|\, \mathbf{T}\big] \tag{2.11}$$

2.1.4 Distorsioni delle lenti

La PPM completa di parametri estrinseci e intrinseci non è sufficiente per modellare correttamente le immagini reali: occorre tenere in conto degli effetti ottici delle lenti presenti nelle camere, in quanto il modello pinhole è un'approssimazione che non le prevede. I parametri aggiuntivi sono non lineari e non alterano il modello di base, ma si applicano seguendo una sequenza specifica:

1. trasformazione dei punti 3D dal sistema WRF al sistema CRF, usando i parametri estrinseci;
2. proiezione prospettica canonica;
3. mapping non lineare della distorsione (*image warping*);
4. mapping dalle coordinate del piano d'immagine alle coordinate in pixel, usando i parametri intrinseci.

Il mapping non lineare prende come variabile indipendente il vettore delle coordinate ideali $[\tilde{x}\ \tilde{y}]^T$ e come variabile dipendente il vettore delle coordinate distorte $[x'\ y']^T$:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} \quad r = \sqrt{(\tilde{x} - \tilde{x}_c)^2 + (\tilde{y} - \tilde{y}_c)^2} \tag{2.12}$$

Dove r è la distanza dal centro di distorsione $(\tilde{x}_c, \tilde{y}_c)$, che per semplicità supponiamo coincidente con il centro d'immagine.
Vengono modellate le due distorsioni maggiori: la distorsione radiale $L(r)$ e la distorsione tangente $[d\tilde{x}\ d\tilde{y}]^T$.

$$L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots \tag{2.13}$$

La distorsione radiale $L(r)$ è approssimata utilizzando la serie di Taylor fino ad un certo ordine (e.g., nelle versioni più recenti di OpenCV, si approssima fino al terzo ordine). La distorsione tangente $[d\tilde{x}\ d\tilde{y}]^T$ invece è approssimata tramite una trasformazione non lineare:

$$\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} = \begin{pmatrix} 2p_1\tilde{x}\tilde{y} + p_2(r^2 + 2\tilde{x}^2) \\ p_1(r^2 + 2\tilde{y}^2) + 2p_2\tilde{x}\tilde{y} \end{pmatrix} \tag{2.14}$$

In conclusione, oltre ai parametri della PPM (i.e., $\mathbf{A}, \mathbf{R}, \mathbf{T}$), occorrono anche i parametri delle distorsioni (i.e., k_1, k_2, p_1, p_2, k_3, secondo l'ordine della libreria OpenCV).

2.1.5 Calibrazione

Una volta definito il modello per definire il processo di acquisizione delle immagini, è necessario stimare tutti i parametri per poter effettuare le elaborazioni necessarie.
Esistono diversi algoritmi, tutti basati su sistemi di equazioni lineari dato una serie di corrispondenze 3D-2D: per ottenere le corrispondenze è necessario un oggetto di calibrazione specifico con delle dimensioni conosciute in modo preciso.
Ogni oggetto di calibrazione ha un pattern conosciuto (e.g., a scacchiera): gli algoritmi di stima possono lavorare su due tipologie di oggetti di calibrazione:

- singola immagine con oggetto di calibrazione con più piani;
- più immagini (anche più di 10, a seconda dell'algoritmo), utilizzando un oggetto di calibrazione con un solo piano.

Data la facilità di costruzione degli oggetti di calibrazione a piano singolo (e.g., foglio di carta), si predilige l'utilizzo di quest'ultimi: in particolare accenneremo al metodo di Zhang [16].

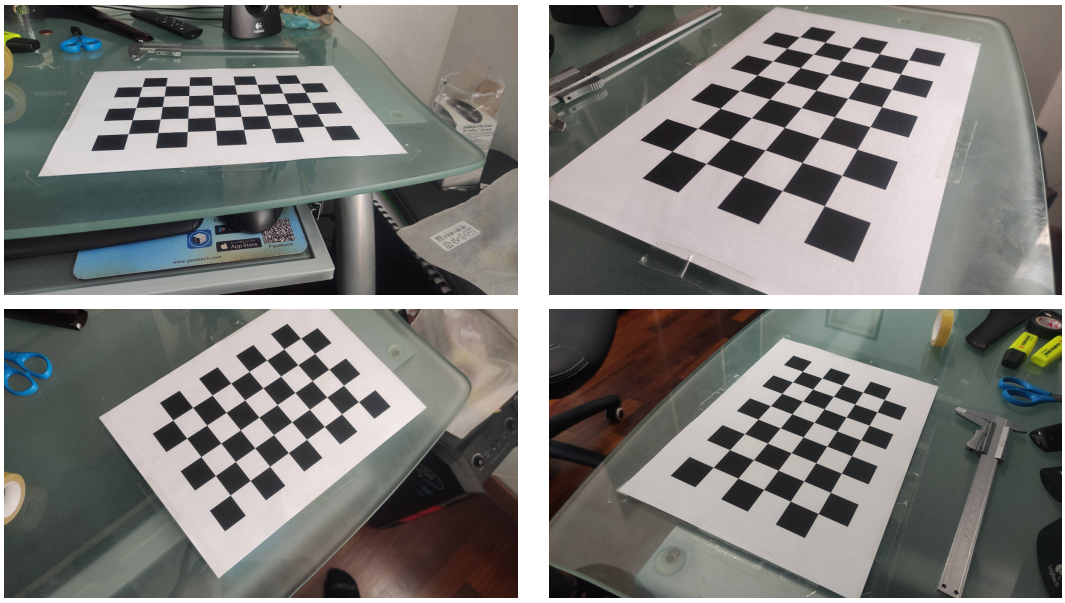


Figura 2.4: calibrazione di un sensore Sony IMX682 tramite metodo di Zhang. Sono state utilizzate più di 20 immagini, ottenendo un errore RMSE di 0.69 px. Si è utilizzato un oggetto di calibrazione a scacchiera misurato con il calibro.

Il metodo di Zhang prevede una serie di passi da svolgere in sequenza per ottenere la stima dei parametri. Come già accennato utilizza un oggetto di calibrazione a scacchiera ed è implementato nelle maggiori librerie disponibili (e.g., OpenCV).

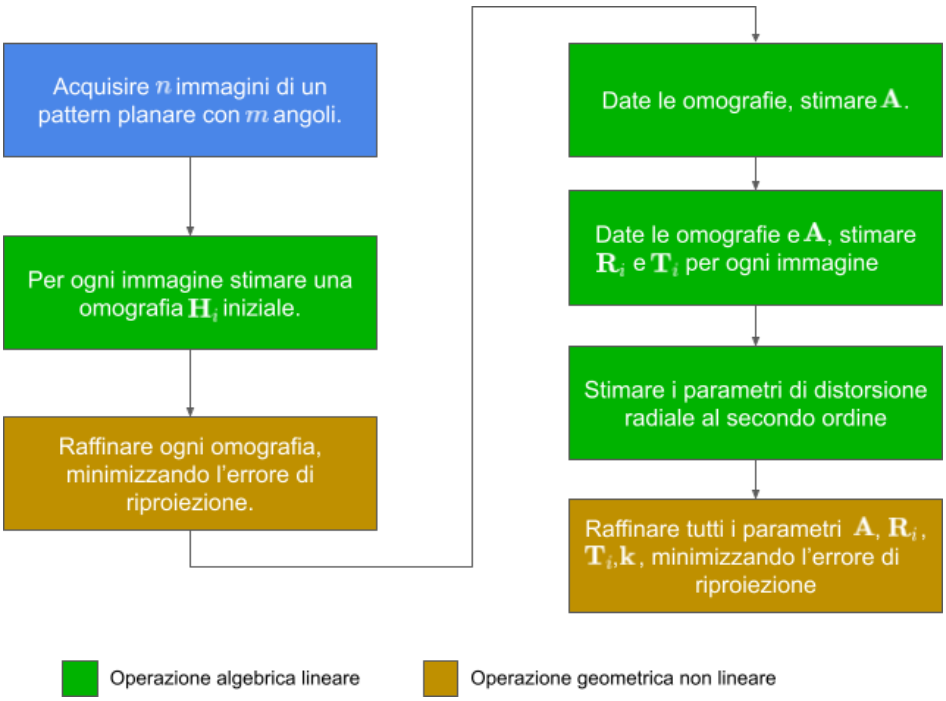


Figura 2.5: schema del metodo di Zhang[4].

L'algoritmo sfrutta ampiamente le omografie (Eq. 2.15), cioè relazioni lineari tra due piani in cui ogni punto di un piano (i.e., la scacchiera) è in relazione con uno e un solo punto nell'altro piano (i.e., il piano d'immagine).

Possiamo ricavare la matrice omografica dalla PPM, se consideriamo solo i punti 3D nel piano della scacchiera ($z = 0$):

$$k\tilde{\mathbf{m}} = \tilde{\mathbf{P}}\tilde{\mathbf{w}} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,4} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H}\tilde{\mathbf{w}}' \qquad (2.15)$$

Le omografie sono stimate tramite sistemi lineari basate su 2.15 (■) e raffinate tramite una minimizzazione dell'errore di riproiezione (■).
Successivamente si stimano tramite sistemi lineari $\mathbf{A}$,$\mathbf{R}$,$\mathbf{T}$ sfruttando i vincoli ortonormali e la 2.15 (■).
Si stimano anche i parametri della distorsione radiale al secondo ordine (■) ed infine si applica una minimizzazione dell'errore di riproiezione (■).

[4]Fonte immagine: [16] (tradotta).

2.2 Sistema stereo laterale

Il sistema stereo laterale è formato da due camere (*reference* e *target*) con una serie di vincoli:

- le camere hanno stessa lunghezza focale f;
- gli assi ottici sono paralleli;
- i piani d'immagine sono coplanari;
- gli assi coordinati sono paralleli.

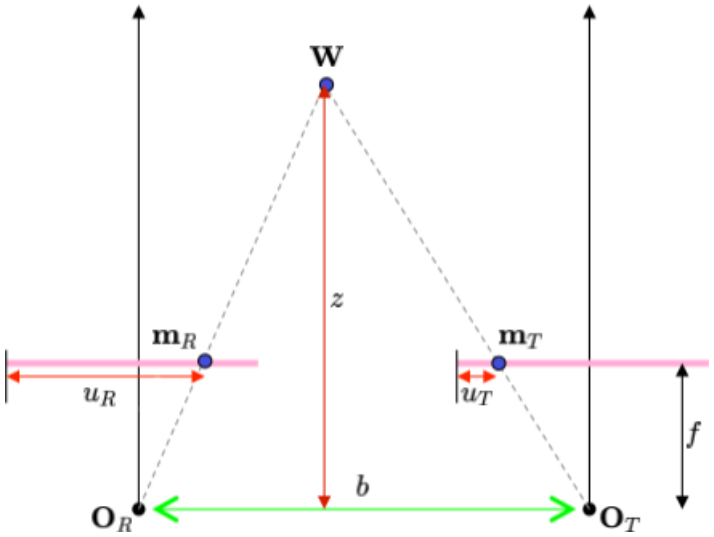


Figura 2.6: modello stereo laterale: le due camere sono poste sullo stesso asse orizzontale[5].

In queste condizioni la rototraslazione per passare dal centro ottico di riferimento ($\mathbf{O}_R$) al centro ottico target ($\mathbf{O}_T$) si semplifica in una traslazione nell'asse x di valore b (baseline). Definiamo le PPM sia per la camera associata al riferimento SRF (di solito la sinistra) sia per la camera target:

$$\tilde{\mathbf{P}}_R = \mathbf{A}\big[\mathbf{I}\,|\,\mathbf{0}\big] \quad \tilde{\mathbf{P}}_T = \mathbf{A}\big[\mathbf{I}\,|\,\mathbf{T}\big] \quad \mathbf{T} = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \tag{2.16}$$

Dato che le due PPM differiscono solo per una traslazione sull'asse x, possiamo facilmente mettere in relazione le coordinate sul piano d'immagine:

[5]Fonte immagine: [2] (adattata).

$$\tilde{\mathbf{m}}_R = \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{z}x_R \\ \frac{f}{z}y_R \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{z}x \\ \frac{f}{z}y \\ 1 \end{bmatrix} = \tilde{\mathbf{P}}_R\tilde{\mathbf{W}} \tag{2.17}$$

$$\tilde{\mathbf{m}}_T = \begin{bmatrix} u_T \\ v_T \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{z}x_T \\ \frac{f}{z}y_T \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{z}(x+b) \\ \frac{f}{z}y \\ 1 \end{bmatrix} = \tilde{\mathbf{P}}_T\tilde{\mathbf{W}} \tag{2.18}$$

Analizzando le equazioni 2.17 e 2.18 possiamo notare che $y_R = y_T = y$ e di conseguenza $v_R = v_T = \frac{f}{z}y$, mentre $x_R - x_T = b$.

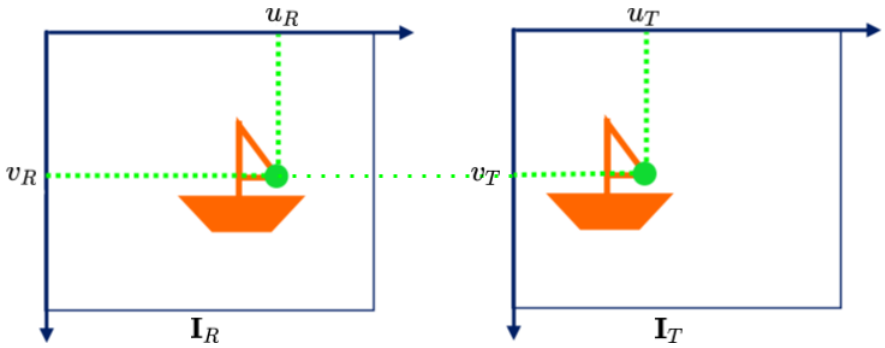


Figura 2.7: immagini fornite dal sistema stereo laterale: i punti corrispondenti differiscono per una traslazione sull'asse x[6].

Possiamo definire la disparità come la differenza tra le coordinate x dei punti corrispondenti nelle due immagini $u_R - u_T = d$, da cui possiamo ricavare la depth z:

$$u_R - u_T = d \quad \rightarrow \quad \frac{f}{z}x_R - \frac{f}{z}x_T = d \quad \rightarrow \quad \frac{f}{z}b = d \quad \rightarrow \quad z = \frac{f}{d}b \tag{2.19}$$

Possiamo codificare la disparità nelle mappe di disparità (Fig. 1.4): sono immagini a scala di grigi dove ogni pixel codifica il valore di disparità in un'intensità luminosa.
Come nel caso delle mappe di profondità, possiamo utilizzare una colormap per fornire una migliore rappresentazione alla disparity map.
Dalla 2.19 notiamo che le distanze sono inversamente proporzionali alle disparità: i colori caldi e/o luminosi rappresentano disparità elevate, ma distanze corte [2, 13, 16].

[6]Fonte immagine: [13] (adattata).

2.2.1 Problema della corrispondenza

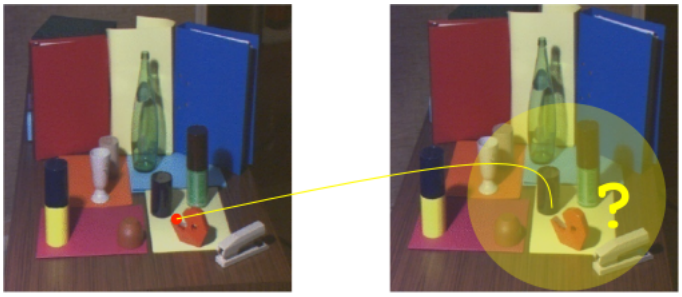


Figura 2.8: problema della corrispondenza in un sistema stereo generico senza vincoli[7].

Per determinare la disparità occorre associare ad ogni punto ($\mathbf{m}_R$) dell'immagine di riferimento il punto corrispondente ($\mathbf{m}_T$) nell'immagine target, cioè la corrispondenza che indicheremo con $\mathbf{m}_R \leftrightarrow \mathbf{m}_T$ [2, 13].

Per semplicità il punto $\mathbf{m}$ sarà riferito alla reference ($\mathbf{m} \equiv \mathbf{m}_R$) e se non specificato in altro modo il riferimento cade sulla camera di sinistra. Anche la disparità $d^*(\mathbf{m}) \equiv d^*_S(\mathbf{m}_R)$ indicherà per semplicità la funzione che mappa i punti $\mathbf{m}_R$ della vista di sinistra nei punti corrispondenti $\mathbf{m}_T$ della vista di destra (Eq. 2.20).

$$\text{(SX)} \quad \mathbf{m} = \mathbf{m}_R = \begin{bmatrix} u \\ v \end{bmatrix} \longleftrightarrow \mathbf{m}_T = \begin{bmatrix} u - d^*(\mathbf{m}) \\ v \end{bmatrix} \quad \text{(DX)} \tag{2.20}$$

Come visto nel Capitolo 1, si tratta di un problema intrinsecamente ambiguo ma che possiamo affrontare mediante l'aggiunta di vincoli (e.g., geometria epipolare, disparità limitata, continuità, unicità e ordinamento) codificati all'interno dell'algoritmo di matching: si tratta dell'algoritmo che genera le mappe di disparità a partire da un paio di immagini stereo rettificate[8].

La tassonomia degli algoritmi prevede due classificazioni: *features-based* e *area-based*.
I primi effettuano il matching solo su un sottoinsieme dei punti (*features*): questi punti hanno delle caratteristiche particolari (e.g., angoli, bordi) che li rendono particolarmente efficaci per il matching. Anche se il matching è veloce e robusto, le mappe generate da questi algoritmi sono sparse. La categoria area-based invece cerca di effettuare il matching per tutti i punti dell'immagine, producendo mappe di disparità dense, tuttavia il risultato è meno robusto.

[7] Fonte immagine: [2].
[8] Vedi sottosezione 2.2.2.

Principali problematiche

Le ambiguità presenti nel problema di corrispondenza derivano principalmente dal modello pinhole, dal modello stereo, dalla scena e dai difetti delle camere reali:

- le occlusioni sono causate dalla diversa prospettiva con cui si osserva la scena: in una vista possono essere presenti punti non visibili nell'altra vista (e.g., dovuti al differente campo visivo delle camere);
- le distorsioni fotometriche sono causate dal rumore, dalle diverse risposte delle camere reali, ma soprattutto dalle superfici non lambertiane della scena: una superficie non lambertiana avrà una diversa luminosità nelle due viste, compromettendo la similarità;
- le distorsioni prospettiche sono causate da regioni non fronto-parallele che appaiono di dimensioni diverse nelle due viste;
- le regioni con scarsa texture o con un pattern ripetitivo rendono i punti difficili da distinguere.

Geometria epipolare

La geometria epipolare permette di ridurre lo spazio di ricerca dei punti corrispondenti da $O(H \times W)$ a $O(W)$, cioè da una ricerca 2D ad una ricerca 1D: tutti i punti da ricercare nel piano d'immagine target sono collocati su delle rette chiamate rette epipolari.

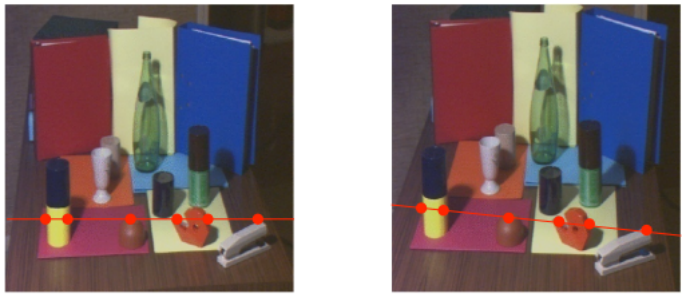


Figura 2.9: problema della corrispondenza in un sistema stereo generico con il vincolo epipolare (rette rosse)[9].

Tutte le rette sul piano d'immagine passano per l'epipolo $(\mathbf{e}_R, \mathbf{e}_T)$: si tratta della proiezione sul piano d'immagine del centro ottico dell'altra camera [2, 13, 15].

[9]Fonte immagine: [2].

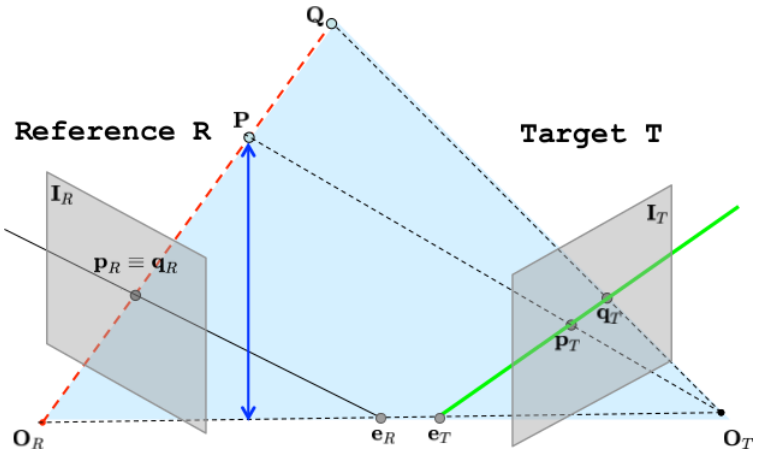


Figura 2.10: il punto $\mathbf{p}$ che giace su $\mathbf{I}_R$ e attraversato dalla retta rossa viene ricercato solo nella retta verde su $\mathbf{I}_T$: le rette $\overrightarrow{\mathbf{O}_R\mathbf{P}}$, $\overrightarrow{\mathbf{O}_T\mathbf{P}}$ e $\overrightarrow{\mathbf{O}_R\mathbf{O}_T}$ sono coplanari ([10]).

Anche se la ricerca dei punti corrispondenti è stata semplificata (Fig. 2.9), risulta comunque più complessa della ricerca in un sistema stereo laterale: in questo sistema le rette epipolari sono coniugate orizzontali e collineari (Fig. 2.11), cioè posso effettuare la ricerca sulla stessa *scanline*.

2.2.2 Rettificazione

Le limitazioni meccaniche presenti nel mondo reale non permettono di costruire un sistema stereo laterale perfetto.
Per aggirare il problema è possibile utilizzare la tecnica della rettificazione, che consiste nel virtualizzare il sistema stereo reale in un sistema stereo laterale perfetto (ai limiti delle distorsioni) tramite *warping* [2].

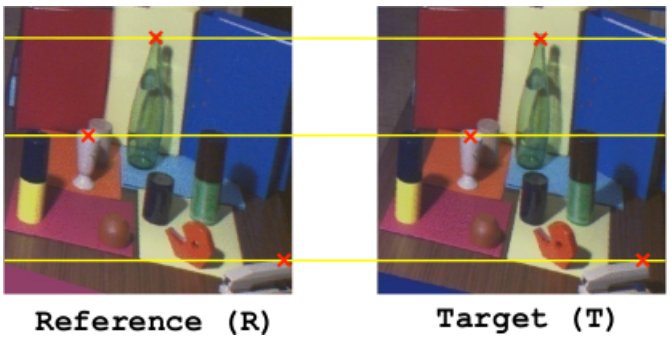


Figura 2.11: problema della corrispondenza in un sistema stereo laterale con vincolo epiolare: dopo la rettificazione le rette epipolari sono coniugate orizzontali e collineari[11].

[10] Fonte immagine: [2].
[11] Fonte immagine: [2].

Prima di iniziare il processo di rettificazione è necessario calibrare il sistema stereo, per esempio mediante una variante del metodo di Zhang, usando SRF come sistema di riferimento esterno. Tuttavia per una maggiore robustezza, dopo aver ottenuto una stima iniziale dei parametri estrinseci $\mathbf{R}$ e $\mathbf{T}$ (e.g., mediana ottenuta dai $\mathbf{R}_i$ e $\mathbf{T}_i$ trovati precedentemente), è necessario minimizzare l'errore geometrico considerando entrambe le camere [16].

Il processo di rettificazione inizia definendo il nuovo sistema stereo laterale, basato sulle componenti del sistema stereo originale $(\mathbf{A}_R, \mathbf{A}_T, \mathbf{R}, \mathbf{T})$ e considerando l'SRF originale come sistema di riferimento esterno [16]:

$$\tilde{\mathbf{P}}'_R = \mathbf{A}'\big[\mathbf{R}' \,|\, \mathbf{0}\big] \quad \tilde{\mathbf{P}}'_T = \mathbf{A}'\big[\mathbf{R}' \,|\, \mathbf{T}'\big] \quad \mathbf{T}' = -\mathbf{R}'\mathbf{O}_T = \mathbf{R}'\mathbf{R}^T\mathbf{T} \tag{2.21}$$

I nuovi parametri intrinseci $(\mathbf{A}')$ sono scelti arbitrariamente (e.g., media tra $\mathbf{A}_R, \mathbf{A}_T$), mentre la matrice di rotazione $(\mathbf{R}')$ deve essere ortonormale e costruita per ottenere un'asse x parallela alla baseline $(\mathbf{B} = \mathbf{O}_T - \mathbf{O}_R = -\mathbf{R}^T\mathbf{T} - \mathbf{0})$ [16].

Si può notare da (2.21) che:

- i parametri intrinseci $(\mathbf{A}')$ sono uguali nelle due PPM permettono di garantire stessa focale f;
- la rotazione $(\mathbf{R}')$ applicata ugualmente nelle due PPM permette di garantire gli assi ottici e gli assi coordinati paralleli;
- la nuova traslazione nella PPM target $(\mathbf{T}' = -\mathbf{R}'\mathbf{O}_T)$, ottenuta ruotando il vecchio centro ottico target $(\mathbf{O}_T = -\mathbf{R}^T\mathbf{T})$, consente di ottenere i piani d'immagine coplanari, garantendo insieme a $\mathbf{A}'$ e a $\mathbf{R}'$ i vincoli del sistema stereo laterale.

Elaborando le vecchie PPM e le nuove PPM possiamo ottenere le omografie di rettificazione:

$$\mathbf{H}_R = \mathbf{P}_R\mathbf{P}'^{-1}_R \quad \mathbf{H}_T = \mathbf{P}_T\mathbf{P}'^{-1}_T \quad \tilde{\mathbf{P}} = \big[\mathbf{P} \,|\, \mathbf{p}_4\big] \tag{2.22}$$

$\mathbf{H}_R$ e $\mathbf{H}_T$ mappano (*image warping*) ciascuna vista in un nuovo spazio in cui le nuove viste sono tra loro rettificate (coppia di immagini rettificate), dopo aver rimosso le distorsioni delle lenti [2, 13, 16]. Va precisato che non esiste una sola coppia di omografie di rettificazione, dato che $\mathbf{A}'$ e $\mathbf{R}'$ hanno dei gradi di libertà: si sceglie la coppia che minimizza la distorsione.

2.3 Stato dell'arte

La letteratura offre da decenni diversi algoritmi di matching (e.g., *Fixed Window (FW)*, SGM, *graph cut*) e ancora oggi si tratta di un argomento con ancora una vasta ricerca. Oltre che all'algoritmo in sé sono state proposte strategie (e.g., *guided stereo*, *spacetime stereo*) a supporto dell'algoritmo. Infine nel Capitolo 4 affronteremo le misure per determinare l'affidabilità di ciascuna corrispondenza trovata.

2.3.1 Tecnologie tradizionali

Le tecnologie tradizionali sono state le prime ad essere utilizzate nel campo: condividono una pipeline comune composta da una serie di passi [2].

1. *Pre-processing*: le immagini subiscono un raffinamento (e.g., *census*, filtro bilaterale) per vincoli della pipeline o per elaborarle più facilmente dopo.

2. *Matching cost computation*: per ogni punto $\mathbf{m}_{Ri}$ sulle rette epipolari orizzontali reference calcolo il costo della possibile corrispondenza di ogni punto $\mathbf{m}_{Tj}$ sulla linea epipolare coniugata target ($\mathbf{m}_{Ri} \tilde{\longleftrightarrow} \mathbf{m}_{Tj}$), producendo il volume di costo (*Disparty Space Image (DSI)*).
   Può essere *pixel-based* o *area-based* (e.g., SAD, SSD): quest'ultima utilizza una finestra di supporto attorno al punto per effettuare la misura.

3. *Cost aggregation*: se si utilizza una finestra di supporto allora si modella la finestra secondo qualche strategia (e.g., *Fixed Window (FW)*, *adaptive window*).
   Le tecniche più sofisticate riescono a ridurre le problematiche sulle corrispondenze[12] [2].

4. *Disparity computation/optimization*: lo scopo dell'algoritmo è trovare la miglior disparità che porti alla giusta corrispondenza per ogni punto.

   - La tipologia di algoritmo locale cerca il valore di disparità usando le informazioni che dipendono solo da un'area vicina al punto (strategia *winner-takes-all*).

   - La tipologia di algoritmo globale invece cerca di determinare la corrispondenza utilizzando tutte le informazioni sull'immagine.

   - Esiste una terza tipologia ibrida che lavora come un algoritmo globale, ma che usa un sottoinsieme delle informazioni sull'immagine.

[12] Già viste in dettaglio nella sottosottosezione 2.2.1.

5. *Disparity refinement*: la mappa di disparità ottenuta è grezza e va raffinata mediante la rimozione di *outliers* (e.g., filtro mediano, misure di confidenza) e l'interpolazione dei dati, per aumentarne la risoluzione.

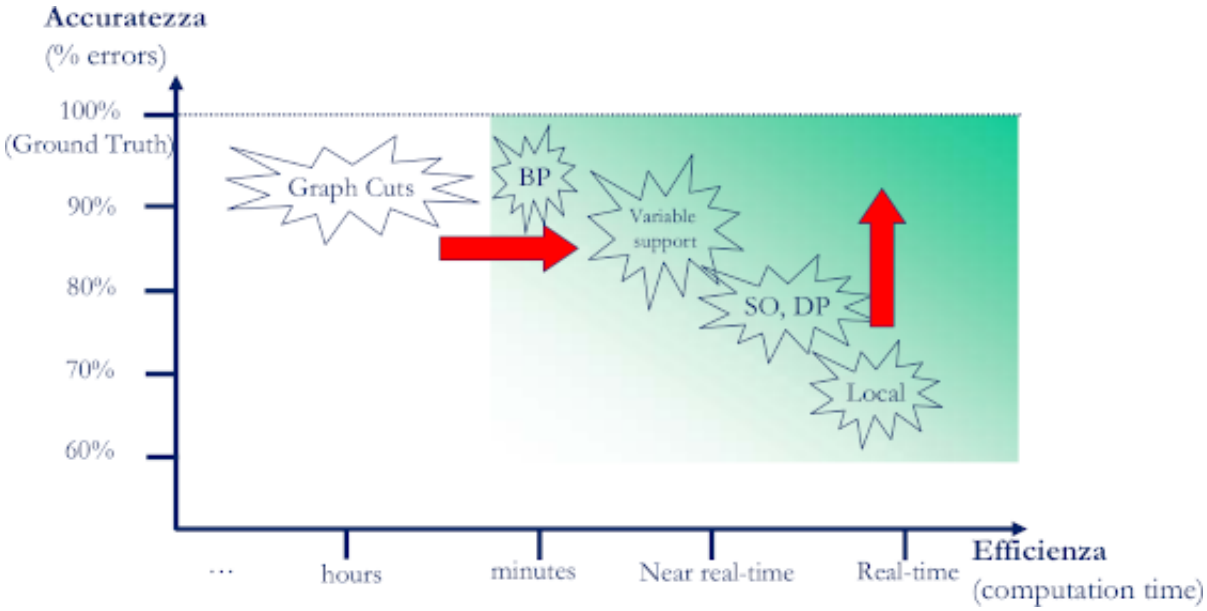


Figura 2.12: comparazione degli algoritmi tradizionali: gli algoritmi locali sono più veloci ma meno robusti[13].

L'algoritmo tradizionale di riferimento utilizzato durante la tesi è *Semi Global Matching (SGM)* [2, 13]: si tratta di una tipologia ibrida con una buona affidabilità e velocità, in cui il problema globale 2D viene scomposto in una serie di problemi 1D definiti su linee (*scanline*) con N direzioni diverse (tipicamente 8 o 16). Per ogni linea si risolve un problema di minimizzazione su una funzione d'energia: il problema è NP-completo, quindi si utilizzano approssimazioni (e.g., *Scanline Optimization (SO)*, *Dynamic Programming*). Nel caso specifico, SGM utilizza SO:

$$L_{\mathbf{r}}(\mathbf{m}, d) = \mathrm{DSI}(\mathbf{m}, d) + \min \begin{cases} L_{\mathbf{r}}(\mathbf{m} - \mathbf{r}, d), \\ L_{\mathbf{r}}(\mathbf{m} - \mathbf{r}, d-1) + P_1, \\ L_{\mathbf{r}}(\mathbf{m} - \mathbf{r}, d+1) + P_1, \\ \min\limits_{i:|d-i|>1} L_{\mathbf{r}}(\mathbf{m} - \mathbf{r}, i) + P_2 \end{cases} \qquad P_1 < P_2 \tag{2.23}$$

- $L_{\mathbf{r}}(\mathbf{m}, d)$ è il costo semi-globale associato alla possibile corrispondenza $\mathbf{m} \equiv \mathbf{m}_R = [u\ v]^T \tilde{\longleftrightarrow} \mathbf{m}_T = [u - d\ v]^T$, ottenuto dall'approssimazione SO nella direzione $\mathbf{r}$;

[13] Fonte immagine: [17].

- DSI($\mathbf{m}, d$) è il termine locale (utilizzato anche negli algoritmi locali) calcolato al passo Matching cost computation e di dimensioni ($H \times W \times D$), con H l'altezza dell'immagine, W la larghezza dell'immagine e con D la dimensione dell'intervallo di disparità $[d_{\min}, d_{\max}]$;
- $\min\{\ldots$ è il termine globale che regolarizza il costo: permette variazioni lente ($+P_1$) (*slanted surfaces*), ma penalizza fortemente i salti di disparità ($+P_2$).

Nelle implementazioni è inserito un termine aggiuntivo di stabilità numerica, ma che non compromette l'ottimizzazione SO.

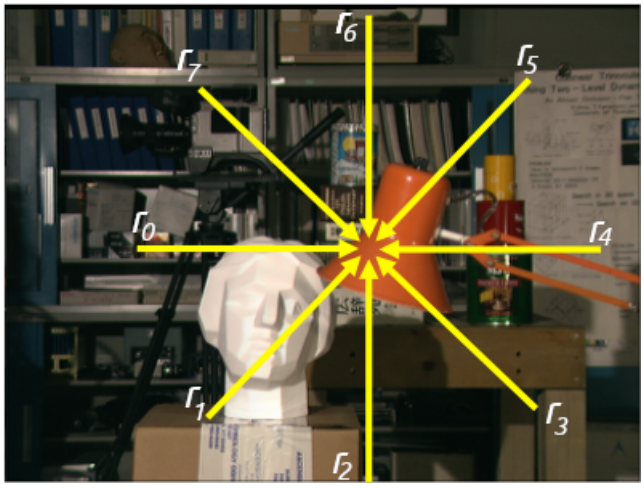


Figura 2.13: direzioni di SO utilizzate in SGM-8[14].

Una volta calcolate tutti i costi semiglobali, si effettua un'aggregazione per mitigare gli *streaking effects* dovuti all'assunzione di indipendenza tra le scanline:

$$S(\mathbf{m}, d) = \sum_{\mathbf{r} \in \{\mathbf{r}_1, \ldots \mathbf{r}_N\}} L_{\mathbf{r}}(\mathbf{m}, d) \tag{2.24}$$

Infine la disparità $d^*(\mathbf{m})$ del punto $\mathbf{m}$ viene scelta con la strategia *winner-takes-all*:

$$\left\{ d^*(\mathbf{m}) \in [d_{\min}, d_{\max}] \middle| d^*(\mathbf{m}) = \arg\min_d S(\mathbf{m}, d) \right\} \tag{2.25}$$

La disparità finale $d^*(\mathbf{m})$ identifica la corrispondenza trovata (Eq. 2.20).

[14]Fonte immagine: [2].

2.3.2 Tecnologie deep learning

Gli algoritmi di stereo matching basati su *deep learning* sono relativamente recenti rispetto alle metodologie tradizionali. La svolta è avvenuta nel 2016 con *Matching Cost - Convolutional Neural Network (MC-CNN)*: si tratta di una modifica alla pipeline tradizionale, applicando una rete neurale convoluzionale profonda alla parte di matching cost computation [18].

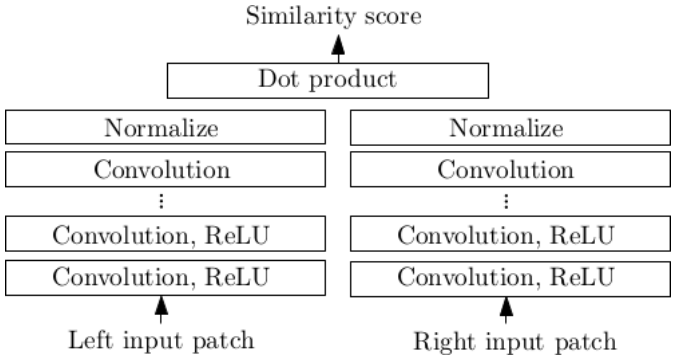


Figura 2.14: architettura MC-CNN nella versione *fast*: si tratta di una rete siamese, con il costo di similarità calcolato tramite prodotto scalare[15].

Gli ultimi progressi nel campo si sono concentrati nel trasformare tutta la pipeline in una rete neurale profonda (*end-to-end stereo*) [19] e di utilizzare architetture fuoriclasse in altre applicazioni (e.g., *optical flow*) [20].

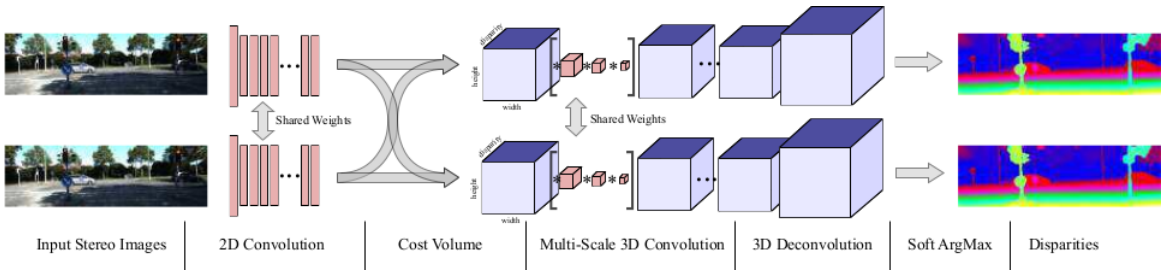


Figura 2.15: architettura *Geometry and Context Network (GC-Net)*: si tratta di una delle prime architetture end-to-end stereo[16].

Tuttavia l'utilizzo di una rete neurale per tutta la pipeline comporta un addestramento molto oneroso in termini di dati: come soluzione sono stati proposti dataset sintetici [21] per un primo addestramento ed infine si effettua un *fine-tuning* nel dominio dell'applicazione.

[15]Fonte immagine: [18].
[16]Fonte immagine: [19].

2.3.3 Stereo matching guidato

Il *domain-shifting* è un problema che affligge le pipeline a rete neurale: il passaggio dal dominio di partenza al nuovo dominio (e.g., la macchina a guida autonoma è entrata in una strada mai vista) comporta una perdita di accuratezza.

Il problema può essere attenuato secondo diversi punti di vista:

1. *self-adaptive stereo*: il passaggio al nuovo dominio è risolto con un fine-tuning continuo, utilizzando una rete leggera e una strategia di *self-learning* online [22];
2. *guided stereo*: le reti neurali (ma applicabile anche i metodi tradizionali) vengono aiutate a capire il nuovo contesto mediante l'utilizzo di informazioni esterne (e.g., sensori attivi): la rete è "abituata" a ricevere informazioni esterne e non è necessario un fine-tuning [6, 7];
3. *self-adaptive stereo* + *guided stereo*: si potrebbe fondere i due metodi precedenti, per esempio utilizzando una strategia di addestramento che consideri le informazioni esterne.

La strategia proposta in questo documento mira a migliorare le metodologie della categoria guided, proponendo un'applicazione a tutti agli algoritmi di matching, con una migliore efficacia rispetto a [7].

2.3.4 Spacetime stereo

Nell'articolo spacetime stereo [8] si è introdotta una metodologia per ridurre gli effetti delle principali problematiche nella corrispondenza (e.g., distorsioni fotometriche, regioni ambigue), mediante un proiettore di pattern: a differenza dei sistemi a luce strutturata tale proiettore non sostituisce una camera nel sistema stereo, bensì viene aggiunto come terzo elemento.

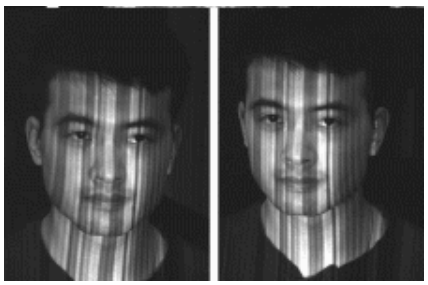


Figura 2.16: esempio di un frame in una sequenza spacetime[17].

[17]Fonte immagine: [8].

Il proiettore cambia nel tempo il pattern proiettato, in modo da evidenziare parti differenti della scena: la pipeline tradizionale è stata modificata per tener conto dei frame con diversa proiezione. Recentemente la tecnica è stata rivisitata utilizzando una pipeline RAFT-Stereo adeguata alla proiezione, per la creazione di un dataset [23].

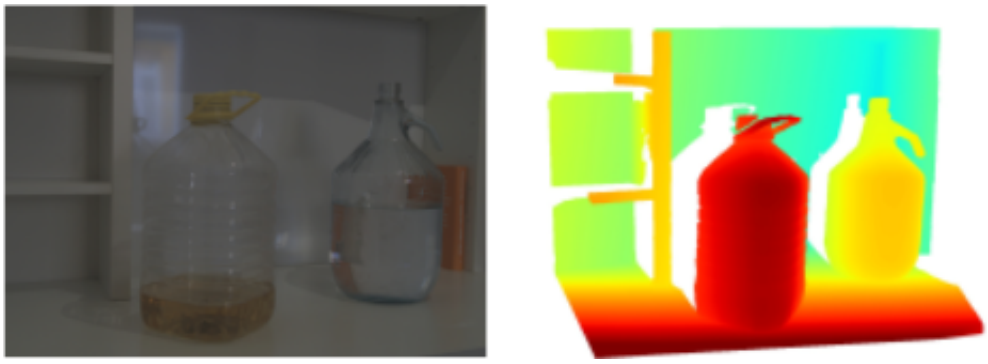


Figura 2.17: gli oggetti composti da vetro sono tutt'oggi difficili per il matching per via della trasparenza e delle superfici non lambertiane[18].

Tuttavia questa tecnica presenta diverse limitazioni intrinseche:

- possono essere catturate con grande affidabilità solo scene indoor, perché fonti di luci esterne potrebbero interferire con la proiezione;
- possono essere catturate solo scene statiche o scene quasi statiche (a seconda del *framerate* massimo gestibile dalla pipeline), altrimenti l'effetto di integrazione nel tempo porta a distorsioni.

La metodologia innovativa descritta nel documento prende ispirazione da [8], tuttavia senza ricorrere ad un vero proiettore: si disegnano i punti affidabili ottenuti da una sorgente esterna direttamente sulle immagini rettificate della scena.

[18]Fonte immagine: [23].

3. Virtual Pattern Projection

Dopo aver introdotto i concetti teorici e lo stato dell'arte nei primi due capitoli, approfondiremo in dettaglio il lavoro svolto: parleremo del ciclo su cui si basa la nuova strategia *Virtual Pattern Projection (VPP)*, delle sue componenti, dei risultati ottenuti e dei suoi limiti.

3.1 Ciclo di proiezione

Il ciclo di proiezione virtuale prevede due fasi che si alternano per un numero limitato di iterazioni: la fase di proiezione, dove i punti vengono proiettati virtualmente in maniera non deterministica, si alterna alla fase di integrazione, dove le immagini modificate vengono aggregate secondo operatori specifici (e.g., media).

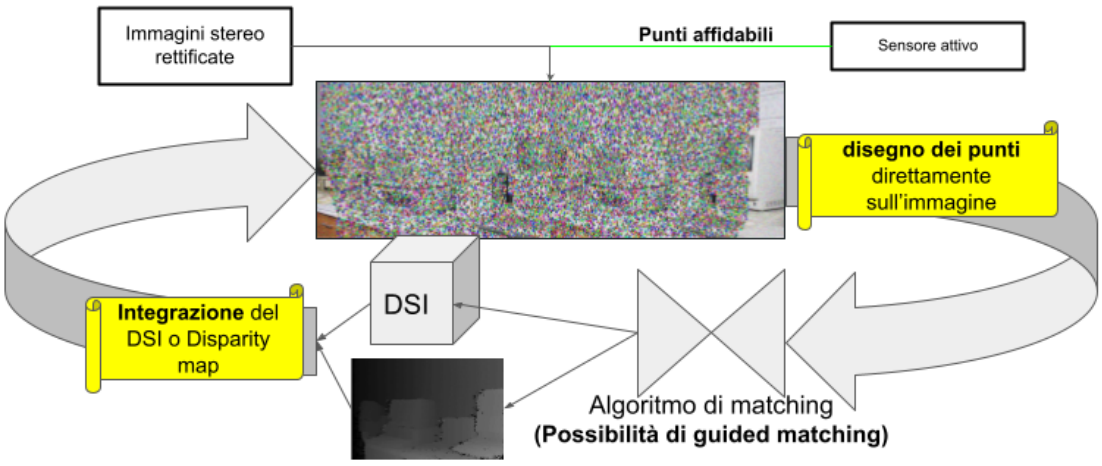


Figura 3.1: schema di funzionamento della strategia VPP: i punti affidabili esterni vengono proiettati sulle immagini stereo per più iterazioni, dopodiché le iterazioni vengono integrate.

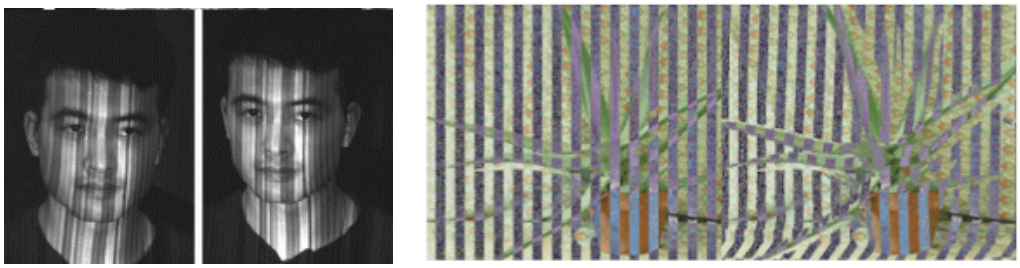
3.1.1 Principio di proiezione

Il principio alla base è molto semplice: conoscendo una serie di punti ricavati da una sorgente esterna $\mathbf{W}_1 \dots \mathbf{W}_n$ (nel sistema di riferimento SRF) e le PPM del sistema stereo laterale, possiamo proiettare un sottoinsieme dei punti su entrambe le immagini rettificate:

$$\begin{bmatrix} \tilde{\mathbf{p}}_{R1} & \dots & \tilde{\mathbf{p}}_{Rn} \end{bmatrix} = \tilde{\mathbf{P}}_R \begin{bmatrix} \tilde{\mathbf{W}}_1 & \dots & \tilde{\mathbf{W}}_n \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} \tilde{\mathbf{p}}_{T1} & \dots & \tilde{\mathbf{p}}_{Tn} \end{bmatrix} = \tilde{\mathbf{P}}_T \begin{bmatrix} \tilde{\mathbf{W}}_1 & \dots & \tilde{\mathbf{W}}_n \end{bmatrix} \tag{3.2}$$

L'intensità luminosa (o i canali RGB se l'immagine è a colori) delle coppie $(\tilde{\mathbf{p}}_{R1}, \tilde{\mathbf{p}}_{T1})$ $\dots$ $(\tilde{\mathbf{p}}_{Rn}, \tilde{\mathbf{p}}_{Tn})$ sarà definita in modo da renderle distinguibili dai punti vicini e identiche tra le due immagini: essendo informazioni ad alta confidenza, si cerca di evidenziarle così da avere un'alta probabilità di corrispondenza. La proiezione può essere eseguita in maniera non deterministica, per esempio aggiungendo rumore e/o modificando ad ogni iterazione il pattern.



proiezione reale[1] proiezione virtuale

Figura 3.2: differenze tra un proiettore reale e una proiezione virtuale che simula il pattern utilizzato in spacetime stereo.

L'utilizzo della proiezione e aggregazione in [8, 23] permette di attenuare alcune ambiguità presenti nella corrispondenza, tuttavia la proiezione virtuale di punti affidabili permette di eliminare quasi del tutto le ambiguità:

- le occlusioni possono essere rimosse portando in visibilità i punti non visibili in tutte le viste;

[1] Fonte immagine: [8].

- le distorsioni fotometriche (i.e., rumore, superfici non lambertiane) non affliggono le proiezioni;
- le distorsioni prospettiche rimangono per via del modello pinhole;
- le regioni con scarsa texture o con pattern ripetitivo vengono arricchite dalla proiezione;
- il proiettore virtuale non è limitato nelle scene outdoor, dato che non viene disturbato da sorgenti luminose esterne.

L'ambiente fisico influenza il proiettore reale usato in [8]: virtualizzando il processo di proiezione si evitano le ambiguità introdotte da esso.

3.1.2 Principio di integrazione

Eseguite m iterazioni, si accumulano i risultati dell'algoritmo di matching (integro nella misura della iterazione):

- se l'algoritmo esplicita un volume dei costi $\mathrm{DSI}(\mathbf{m}, d)$ (i.e., modello *white-box*), allora lo si aggregherà in un volume dei costi integrato (*integrated Disparty Space Image (iDSI)*);

$$\mathrm{IDSI}(\mathbf{m}, d) = F(\mathrm{DSI}_0(\mathbf{m}, d), \ldots, \mathrm{DSI}_m(\mathbf{m}, d)) \tag{3.3}$$

- se l'algoritmo non esplicita un volume dei costi (i.e., modello *black-box*), allora si aggregherà la mappa di disparità.

$$\mathrm{IMAP}(\mathbf{m}) = F(d_0^*(\mathbf{m}), \ldots, d_m^*(\mathbf{m})) \tag{3.4}$$

Dove $F(\cdot)$ è la funzione di aggregazione adottata (e.g., media, somma).

Durante l'elaborato si è utilizzato SGM come algoritmo di matching, di conseguenza è stato implementato solo l'aggregazione del volume dei costi. Per aggregare le informazioni possiamo fare affidamento a vari operatori, tuttavia è stato scelta la media per la stabilità numerica.

Anche in [8, 23] esiste una integrazione, tuttavia viene eseguita nella misura del tempo: ciò limita l'applicazione alle sole scene statiche o quasi statiche (Fig. 3.3).

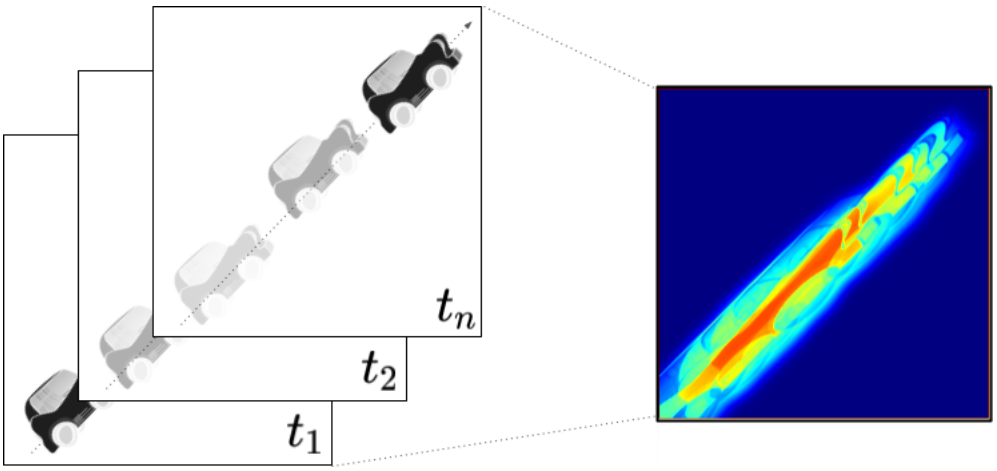


Figura 3.3: se la scena non rimane stabile per un tempo sufficiente allora la mappa di disparità viene distorta dall'integrazione nel tempo in [8] (*motion blur*).

Invece se integriamo nelle iterazioni, la scena è intrinsecamente statica: il pattern viene applicato ad una sola coppia di immagini (Fig. 3.4).

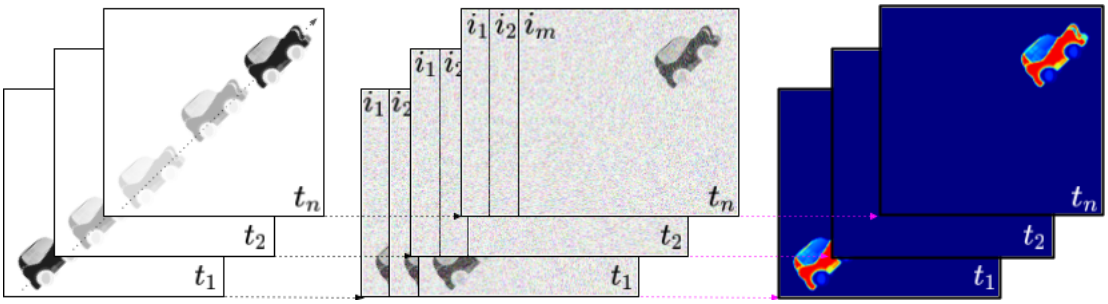


Figura 3.4: anche se la scena è dinamica, ogni ciclo di proiezione virtuale mantiene la stessa immagine.

Va precisato però che integrare nel tempo porta anche ad alcuni vantaggi:

- le distorsioni fotometriche possono essere ridotte, in particolare il rumore tramite un filtro temporale[2] e i riflessi dovuti alle superfici non lambertiane vengono attenuati cambiando pattern;
- le regioni con scarsa texture vengono arricchite dalla successione di pattern (in modo simile ai codewords basati sul tempo dei sistemi a luce strutturata).

[2] Argormento approfondito nella sottosottosezione 3.2.5.

3.2 Pipeline

Una volta introdotto il principio, possiamo passare ai dettagli dell'implementazione: la pipeline stereo tradizionale di partenza[3] è basata su *rapid Semi Global Matching (rSGM)* [24].

Lo sviluppatore ha implementato la pipeline stereo con le seguenti caratteristiche:

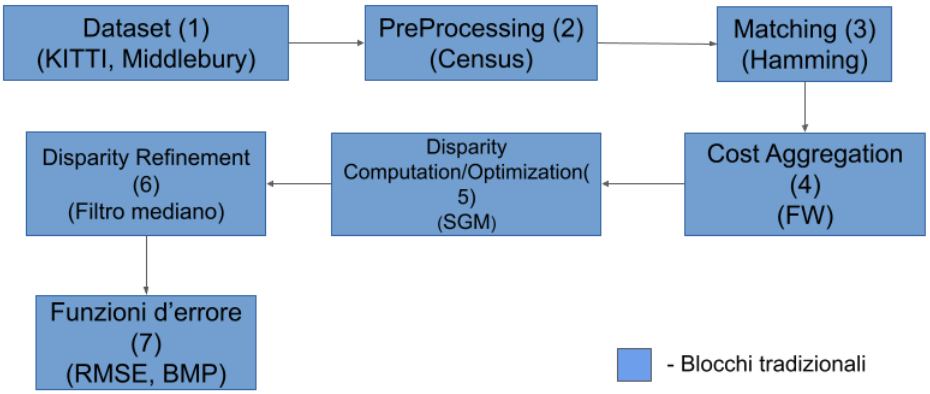


Figura 3.5: pipeline tradizionale nella sua implementazione rSGM originale.

2. pre-processing basato sulla trasformata census[4];
3. funzione di costo basata sulla distanza di Hamming[5];
4. aggregazione del costo mediante finestra fissa (FW) 5×5, implementata intrinsecamente nella trasformata census;
5. ottimizzazione delle disparità mediante metodo SGM a 8 vie;
6. raffinazione delle disparità mediante filtro mediano;

- utilizzo delle istruzioni *Single Istruction Multiple Data (SIMD)* per velocizzare l'algoritmo.

Per implementare il principio, la pipeline originariamente implementata è stata modificata per contenere i blocchi aggiuntivi che gestiscono il principio di proiezione e il principio di integrazione:

[3]Link al sorgente: `https://github.com/ivankreso/stereo-vision`
[4]Descritta in dettaglio nella sottosezione 3.2.3.
[5]Anche la distanza di Hamming è descritta in dettaglio nella sottosezione 3.2.4.

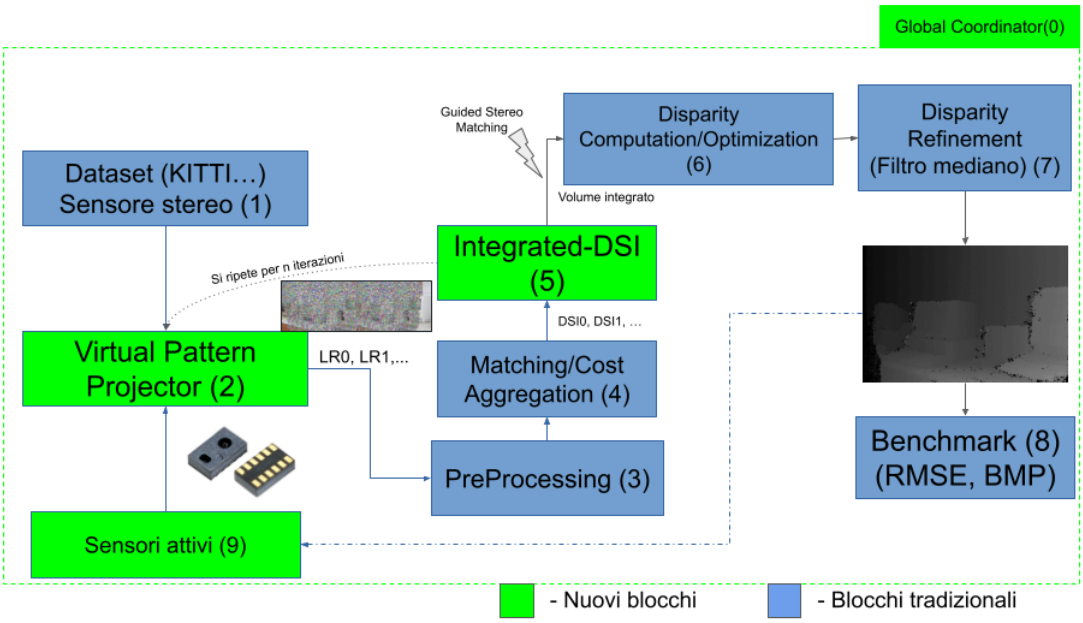


Figura 3.6: pipeline rivisitata con l'aggiunta dei componenti per il ciclo di proiezione.

0. coordinatore globale: data l'interdipendenza tra i vari blocchi, si è deciso di sviluppare un coordinatore che gestisca globalmente il contesto;
2. proiettore virtuale: riceve le immagini rettificate dal sensore stereo o dal dataset, i punti affidabili dal sensore attivo e produce le immagini con il pattern, secondo i parametri inseriti;
5. volume del costo integrato: aggrega i volumi di costo ricevuti dall'algoritmo ad ogni iterazione, usando la media;
9. Sensori attivi: permettono di ottenere i punti affidabili (nel lavoro svolto sono stati usati i punti forniti dal dataset).

Per ogni blocco vedremo gli iperparametri e i dettagli implementativi tra cui le scelte di ottimizzazione e l'interdipendenza tra i blocchi. Tutti i blocchi sono stati implementati mediante il linguaggio Python 3: sono state utilizzate le librerie integrate, ma anche NumPy, Cython, OpenCV, Tensorflow, Matplotlib, Ray e altre.

3.2.1 Coordinatore globale

Il coordinatore globale è l'entità che inizializza i blocchi della pipeline, definisce i collegamenti tra gli output di un blocco e l'input di un altro (*flow*) e distribuisce informazioni aggiuntive (contesto) in blocchi anche non adiacenti.

La definizione del flow avviene staticamente prima del ciclo di proiezione, in base ai blocchi presenti.

Iperparametri	Tipo	Default	Descrizione
`iterations`	intero	10	indica il numero di iterazioni da eseguire

Tabella 3.1: iperparametri principali del coordinatore globale.

Il contesto può essere utile al singolo blocco per tenere traccia del ciclo (e.g., numero di iterazioni mancanti).
Una volta finito il ciclo di proiezione, il coordinatore restituisce i risultati tramite una *callback*.

3.2.2 Dataset

Per addestrare le nuove metodologie del settore e verificarne l'efficacia si utilizzano i *dataset*: da una serie di scene (indoor, outdoor) vengono ricavati dati, composti da coppie di immagini rettificate e mappe di disparità associate alla singola coppia (*ground-truth*).

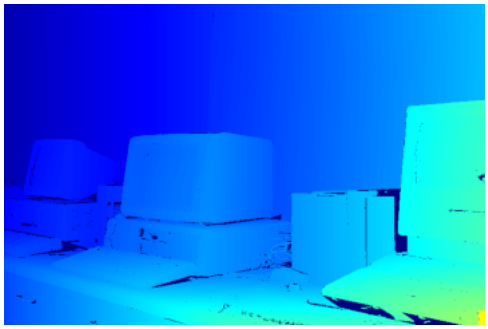
Il blocco dataset permette la lettura e la pre-elaborazione dei dataset, mediante l'utilizzo degli opportuni iperparametri: le immagini possono essere scalate e tagliate per velocizzare la fase di matching. Inoltre si possono saltare delle scene, così una eventuale ricerca di iperparametri o valutazione sarà più veloce.

Le mappe di disparità fornite sono molto affidabili e acquisite con tecniche solitamente attive [5, 25, 26], ma esistono dataset che utilizzano sistemi ibridi (passivi+attivi) per ricavare le mappe [23].

Durante il lavoro svolto sono stati utilizzati sia dataset per ambienti indoor (i.e., Middlebury 2014 [5], nella versione "MiddEval3"), sia dataset per ambienti outdoor (i.e., KITTI12 [25] e KITTI15 [26]).

Il dataset Middlebury2014 fornisce scene costruite appositamente per stressare l'algoritmo di matching. Le mappe sono dense e molto accurate, ottenute mediante tecnica a luce strutturata: per verificare e migliorare l'accuratezza dei dati raccolti sono stati eseguite calcoli (e.g., *plane fitting*) e usate tecniche di *post-processing* (e.g., rimozione di outliers, filtri) [5].

ambiente indoor

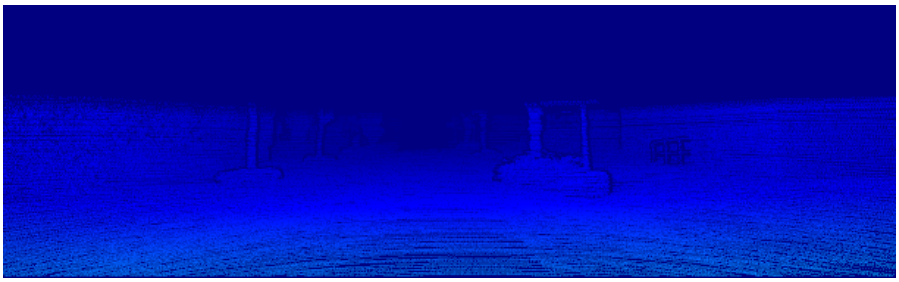


mappa di disparità densa

Figura 3.7: scena Vintage nel dataset Middlebury 2014: si può notare che l'ambiente è di tipo indoor con molte ambiguità e che le disparity map sono dense[6].

Il dataset KITTI12 e KITTI15 fornisce scene ottenute sul campo: un'autovettura è stata dotata di un sensore stereo per raccogliere le immagini rettificate e di un sensore attivo LiDAR per la generazione delle mappe di disparità. Anche in questi dataset sono state usate tecniche di post-processing, soprattutto per la rimozione di outliers e per aumentare la densità delle mappe sparse [25, 26].

ambiente outdoor



mappa di disparità sparsa

Figura 3.8: scena 000000_10 nel dataset KITTI12: si può notare che l'ambiente è di tipo stradale e le disparity map sono sparse[7].

[6]Fonte immagini: [5] (disparity map con colorazione jet).
[7]Fonte immagini: [25] (disparity map con colorazione jet).

Iperparametri	Tipo	Default	Descrizione
`scaleFactor`	$[0.0, 1.0]$	1.0	permette di scalare le dimensioni delle immagini per velocizzare l'esecuzione
`gtDensity`	$[0.0, 1.0]$	0.05	campiona uniformemente la ground-truth dal 0% al 100%
`npairs`	intero	-1	indica il numero di scene da leggere (con -1 legge tutte le scene)
`index`	intero	0	indica da quale scena partire
`crop_for_rsgm`	booleano	vero	effettua un taglio della lunghezza (w) tale che $w \mod 16 = 0$[8]
`useRGB`	booleano	falso	per leggere l'immagine a colori o a scala di grigi

Tabella 3.2: iperparametri principali del blocco dataset.

Non sono state necessarie ottimizzazioni particolari, ma solo le librerie sopra citate.

3.2.3 Pre-processing

Lo stadio di pre-elaborazione permette di effettuare delle trasformazioni alle immagini originali, principalmente per ridurre gli effetti fotometrici e quindi consentire un miglior matching [2]. Le tecniche utilizzate sono molte(e.g., filtro bilaterale, LoG, sottrazione della media), però vedremo solo la trasformata di census dato che è utilizzata nell'implementazione di rSGM.

La trasformata di census utilizza immagini a scala di grigi e prevede una finestra di supporto $\mathcal{N}_{\mathbf{m}}$ che permette di trasformare i pixel della finestra $\mathcal{N}_{\mathbf{m}}$ in stringhe di bit, mediante una comparazione con il pixel centrale $\mathbf{m} = [u\ v]^T$:

$$C(\mathbf{m}) = \xi(\mathbf{m}, \mathbf{m}_1) \otimes \xi(\mathbf{m}, \mathbf{m}_2) \otimes \cdots \otimes \xi(\mathbf{m}, \mathbf{m}_n) \quad \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n \in \mathcal{N}_{\mathbf{m}}$$
$$\xi(\mathbf{m}, \mathbf{m}') = \begin{cases} 1 & \text{se } I(\mathbf{m}) \leq I(\mathbf{m}') \\ 0 & \text{se } I(\mathbf{m}) > I(\mathbf{m}') \end{cases} \tag{3.5}$$

Con $\otimes$ intendiamo l'operatore di concatenazione di bit, con $I(\mathbf{m})$ intendiamo l'intensità dell'immagine a scala di grigi nel punto $\mathbf{m}$ e inoltre i punti della finestra $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n$ devono essere sempre stesso ordine (Fig. 3.9, in alto a destra).

[8] Il vincolo è dato dall'implementazione di SGM utilizzata.

Il risultato della trasformazione di census è una stringa di bit (Fig. 3.9), con una lunghezza n che dipende dalla dimensione della finestra di supporto (e.g., una finestra 3×3 produce una stringa da 8 bit, mentre una finestra 5×5 produce una stringa da 24 bit).

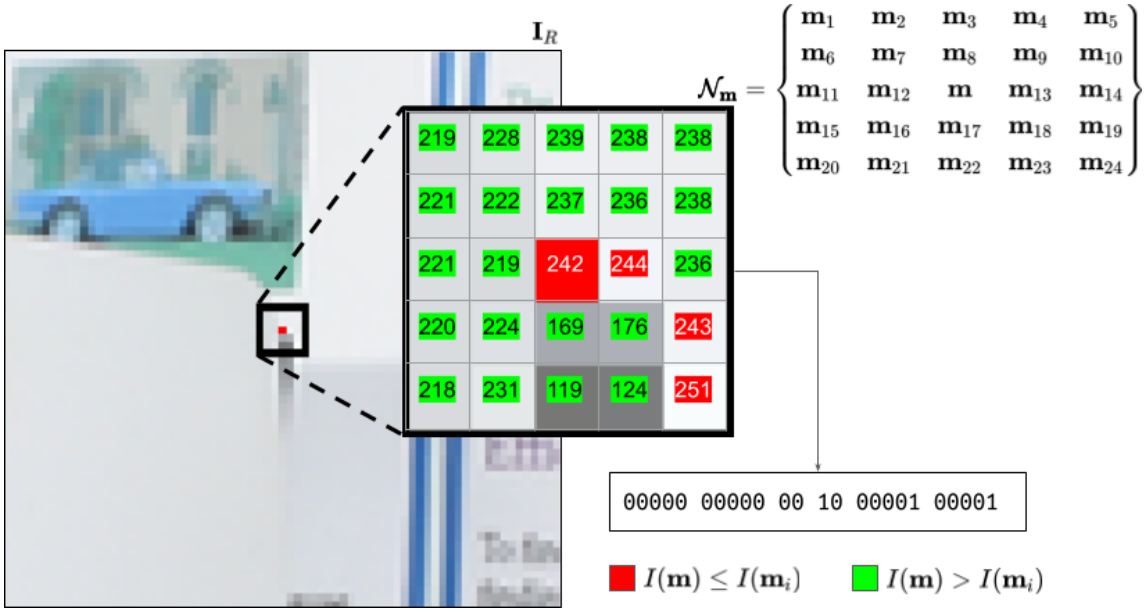


Figura 3.9: esempio di trasformata di census usando una finestra di supporto 5×5: l'ordine dei pixel della finestra è riga per riga, dall'alto verso il basso[9].

Il vantaggio della trasformata di census è la velocità di computazione e il ridotto impatto in memoria, inoltre permette di ridurre le distorsioni fotometriche.

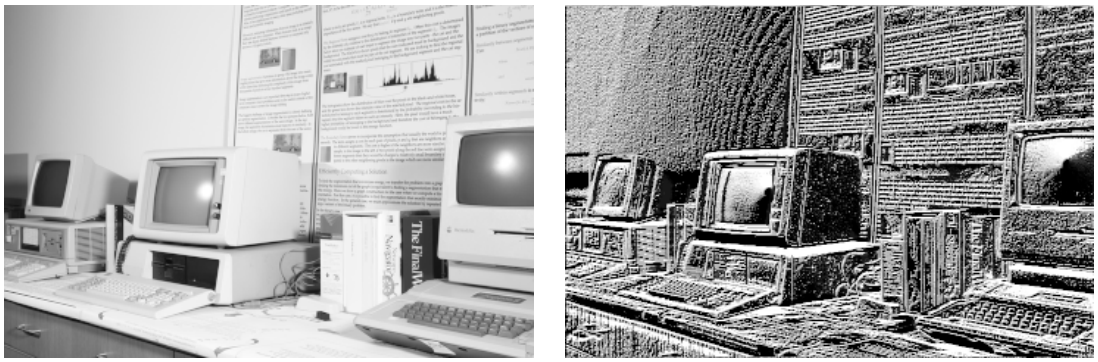


Figura 3.10: trasformata di census sull'immagine Vintage - Middlebury 2014[10].

Lo stadio è stato ottimizzato dallo sviluppatore originale con le istruzioni SIMD e non possiede iperparametri.

[9] La sezione di $\mathbf{I}_R$ è presa da Vitange - Middlebury 2014 [5].
[10] Fonte immagine di partenza: [5].

3.2.4 Matching

Lo stadio di matching è stato implementato nella libreria rSGM usando la distanza di Hamming: date due stringhe di bit C, C' (i.e., trasformate di census), calcoliamo la distanza di Hamming come:

$$\text{Hamm}(C, C') = \sum_{i=1}^{L} (1 - \delta_{C_i C'_i}), \quad \delta_{ij} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \tag{3.6}$$

Indichiamo con δ_{ij} il delta di Kronecker e con L la lunghezza delle stringhe C e C'. Inoltre, con C_i e C'_i indichiamo il i-esimo bit rispettivamente della stringa C e C'.

$$\begin{aligned} C &= 00000\,00000\,00\,10\,00001\,00001 \\ C' &= 00111\,00111\,00\,11\,00000\,00000 \\ &\text{Hamm}(C, C') = 9 \end{aligned} \tag{3.7}$$

Possiamo quindi definire il volume dei costi a partire dalla distanza di Hamming, il quale definisce le probabili corrispondenze:

$$\begin{aligned} &\text{DSI}(\mathbf{m}, d) = \text{Hamm}(C_R(\mathbf{m}_R), C_T(\mathbf{m}_{Td})) \\ &\mathbf{m} \equiv \mathbf{m}_R = \begin{bmatrix} u \\ v \end{bmatrix} \tilde{\longleftrightarrow} \mathbf{m}_{Td} = \begin{bmatrix} u - d \\ v \end{bmatrix} \end{aligned} \tag{3.8}$$

Iperparametri	Tipo	Default	Descrizione
`dmin`	$[0, \dots [$	0	disparità minima calcolata durante il matching
`dmax`	$]\text{dmin}, \dots [$	128	disparità massima calcolata durante il matching

Tabella 3.3: iperparametri principali del blocco matching: `dmin` e `dmax` formano la dimensione del volume di costo.

Il vantaggio del matching tramite distanza di Hamming è la possibilità di parallelizzazione tramite istruzioni SIMD (utilizzate nell'implementazione rSGM) e la conseguente riduzione dei tempi di calcolo.

3.2.5 Proiezione virtuale

Il blocco di proiezione virtuale è la colonna portante dell'intero lavoro: esegue la proiezione del pattern sulle immagini, alterandole virtualmente. Si tratta del blocco con più iperparametri in assoluto: si è cercato di sviscerare il più possibile l'argomento, modellando più aspetti possibili.

Gli aspetti principali su cui si basa la proiezione sono: pattern spaziale, pattern cromatico, rumore aggiunto e dimensione dei punti sull'immagine.

Pattern spaziale

Dato che la posizione dei punti affidabili è vincolata dal modello pinhole (Eq. 3.1, Eq. 3.2), il pattern spaziale è deciso da quali e quanti punti appariranno sulle immagini. Descriveremo i pattern spaziali di tipo casuale, a strisce e a scansioni, in ordine di ideazione e implementazione.

Pattern spaziale identità Non vengono proiettati i punti affidabili: serve per testare il rumore Gaussiano e il blocco di integrazione.

Pattern spaziale casuale I punti affidabili vengono campionati uniformemente: si può decidere la percentuale di punti da inserire nello schermo.

Figura 3.11: pattern spaziale casuale con 6 iterazioni (vista da camera reference).

Pattern spaziale a strisce Viene esteso il pattern utilizzato dal proiettore in Spacetime stereo [8]: si tratta di una successione di 4 pattern ripetitivi a strisce verticali, tuttavia il pattern è stato esteso ad un numero arbitrario di strisce ed è stata aggiunta una traslazione orizzontale del pattern ad ogni iterazione.

	Intensità striscia															
$i = 0$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$i = 1$	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
$i = 2$	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0
$i = 3$	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1

Tabella 3.4: pattern luminoso originale utilizzato in Spacetime stereo.

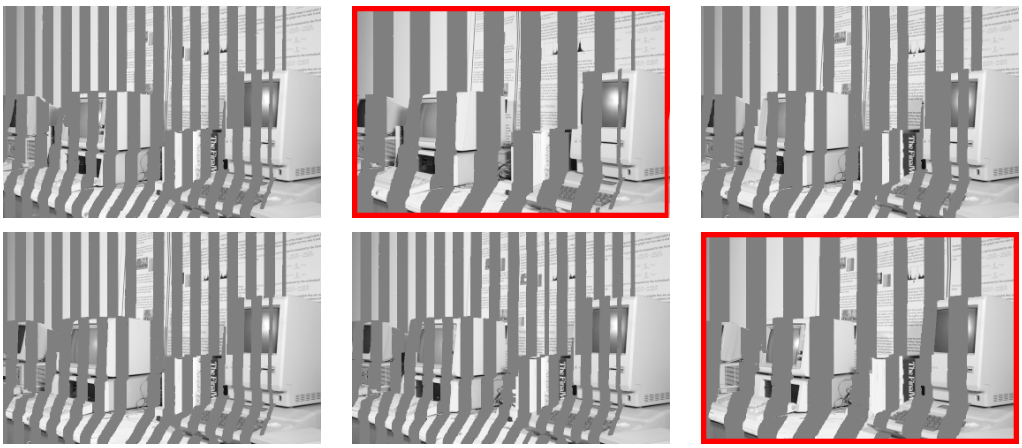


Figura 3.12: pattern spaziale a strisce con 6 iterazioni (vista da camera target): si può notare nelle immagini evidenziate di rosso la traslazione orizzontale.

Pattern spaziale a scansioni Tutti i punti affidabili disponibili vengono disegnati sullo schermo: i punti vengono disegnati nella direzione dell'asse u, cambiando ad ogni iterazione il verso, cioè i punti vengono alternativamente disegnati da sinistra a destra e da destra a sinistra (con riferimento al piano d'immagine $\mathbf{I}_R$).

Figura 3.13: pattern spaziale a scansioni con 2 iterazioni (vista da camera target): si può notare che le parti occluse vengono portate in primo piano in certe iterazioni.

Il pattern a scansione ha un duplice vantaggio:

- può eliminare le occlusioni fino a due livelli (*foreground*, *background*), a patto di possedere punti affidabili sia in primo piano sia in secondo piano (Fig. 3.14): se la scansione è da sinistra a destra appariranno gli oggetti in foreground, mentre se la scansione è da destra a sinistra appariranno gli oggetti in background (considerando come reference la camera di sinistra);

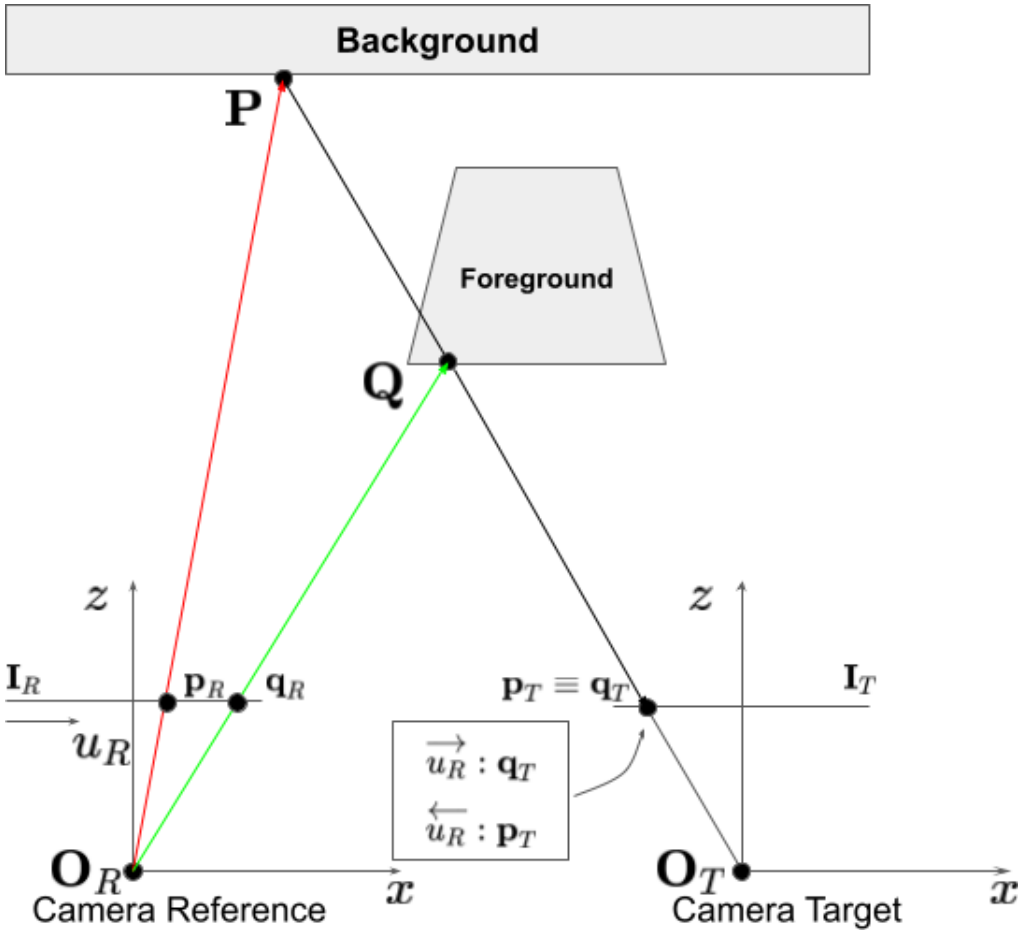


Figura 3.14: i punti vengono disegnati a partire dal piano d'immagine reference ($\mathbf{I}_R$): applicando la proiezione da sinistra a destra ($\overrightarrow{u_R}$), il punto $\mathbf{q}_R$ è disegnato dopo il punto $\mathbf{p}_R$, di conseguenza $\mathbf{q}_T$ coprirà il punto $\mathbf{p}_T$ precedentemente disegnato, mentre se applichiamo la proiezione da destra a sinistra ($\overleftarrow{u_R}$) allora $\mathbf{p}_T$ coprirà $\mathbf{q}_T$.

- permette di disegnare tutti i punti affidabili contemporaneamente: secondo i test effettuati è più efficace, anche il pattern cromatico verrà stressato maggiormente.

Il pattern spaziale è stato ottimizzato mediante Cython per velocizzare il blocco: tutte le modalità sono incluse nell'ottimizzazione.

Iperparametri	Tipo	Default	Descrizione
`class`	{"identity", "scan", "rnd", "striped"}	"scan"	seleziona la tipologia di pattern spaziale
`density`	$[0.0, 1.0]$	1.0	Imposta la densità di campionamento dei punti affidabili dal 0% al 100%
`nstripes`	$[8, 16, 24, \dots]$	16	indica il numero di strisce verticali
`hoffset`	$[0.0, 1.0]$	0.1	indica la traslazione sull'asse x delle strisce, in proporzione alla lunghezza dell'immagine (dal 0% al 100%)

Tabella 3.5: iperparametri principali del pattern spaziale.

Pattern cromatico

Il pattern cromatico si occupa di rendere distinguibili i punti proiettati dai punti originali dell'immagine e di rendere uguali i punti proiettati nelle due viste: lo scopo è un'alta probabilità di matching. Assumeremo immagini a scala di grigi per semplicità, ma i pattern possono essere estesi anche per immagini a colori.

Dato un punto affidabile $\mathbf{P}$ e le sue proiezioni sui piani d'immagine $\mathbf{p}_R$ e $\mathbf{p}_T$, per ottenere un buon matching sarà necessario un'uguaglianza nel dominio delle intensità dei grigi $I_R(\mathbf{p}_R) = I_T(\mathbf{p}_T) = I(\mathbf{p}) \in [0, L[$ (per semplicità omettiamo R e T sia per distinguere l'intensità delle immagini sia per distinguere i punti sui piani d'immagine), oppure nel dominio della trasformata di census $C_R(\mathbf{p}_R) = C_T(\mathbf{p}_T) = C(\mathbf{p})$ (anche in questo caso omettiamo R e T), vincolando lo stadio di pre-processing con una trasformata di census.

Descriveremo pattern cromatici nel dominio dei colori (i.e., casuale, negativo, a distanza massima), ma anche nel dominio della trasformata di census (i.e., "census" e "census2").

Pattern cromatico negativo Viene effettuata una media su una finestra di supporto ($\mathcal{N}_R$) attorno al punto $\mathbf{p}_R$ (trascurando $\mathcal{N}_T$ perché assumiamo sia simile a $\mathcal{N}_R$):

$$\bar{I}(\mathbf{p}_R) = \frac{1}{|\mathcal{N}_R|} \sum_{\mathbf{q} \in \mathcal{N}_R} I_R(\mathbf{q}) \tag{3.9}$$

Il valore cromatico di proiezione $I(\mathbf{p})$ è il negativo di $\bar{I}(\mathbf{p}_R)$:

$$I(\mathbf{p}) = (L-1) - \bar{I}(\mathbf{p}_R) \tag{3.10}$$

Considerando $L-1$ come valore massimo di intensità.

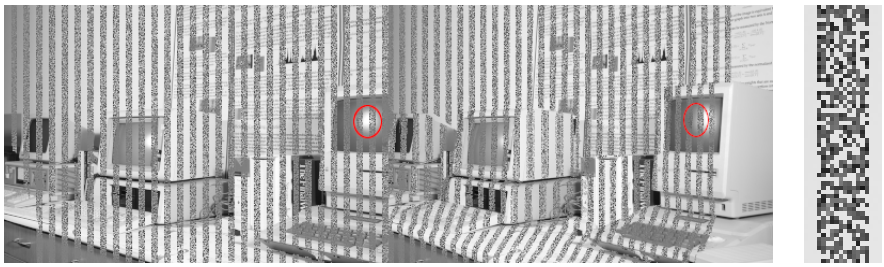


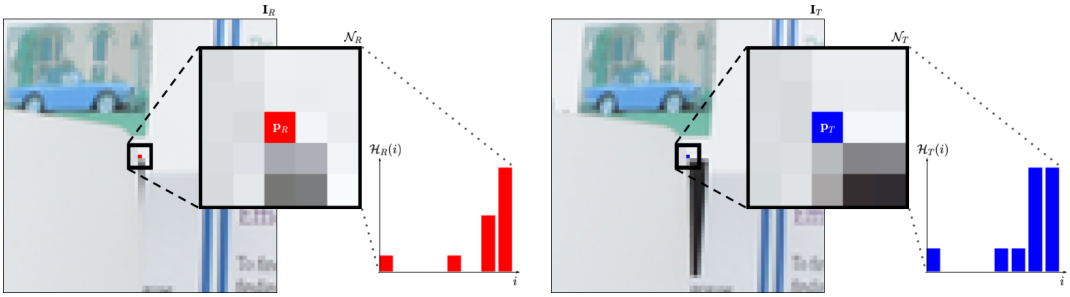
Figura 3.15: pattern cromatico negativo e pattern spaziale a strisce, con vista reference (sinistra) e target (destra): si può notare la granularità delle strisce (ingrandimento a destra) e l'effetto nelle zone non lambertiane (cerchio rosso).

La modifica dell'intensità $I_R(\mathbf{p}_R)$ comporta un *side-effect* sul calcolo delle medie successive nell'intorno di $\mathbf{p}_R$: come si può notare dalla figura 3.15, le strisce sono granulari. Inoltre trascurando $\mathcal{N}_T$, il pattern perde d'efficacia nelle superfici non lambertiane.

Pattern cromatico a distanza massima Sono previste due finestre di supporto ($\mathcal{N}_R$ e $\mathcal{N}_T$) attorno ai punti $\mathbf{p}_R$ e $\mathbf{p}_T$: le finestre servono per creare un istogramma basato su entrambe le finestre.

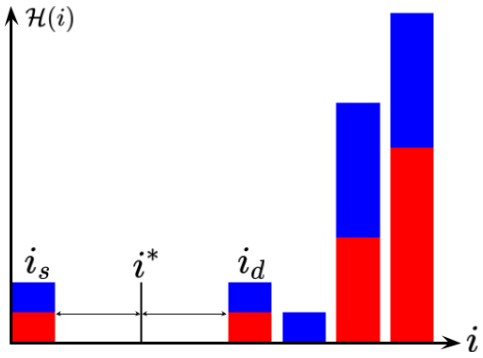
$$\mathcal{H}(i) = \mathcal{H}_R(i) + \mathcal{H}_T(i) = \sum_{\mathbf{q}\in\mathcal{N}_R} \delta_{iI_R(\mathbf{q})} + \sum_{\mathbf{q}\in\mathcal{N}_T} \delta_{iI_T(\mathbf{q})} \tag{3.11}$$

Dove $\mathcal{H}(i)$ è la funzione d'istogramma al variare dell'intensità $i \in [0, L[$ e δ_{ij} è il delta di Kronecker.



Vista reference, $\mathcal{H}_R(i)$ Vista target, $\mathcal{H}_T(i)$

Figura 3.16: costruzione dell'istogramma a partire dalle due finestre di supporto $\mathcal{N}_R$ e $\mathcal{N}_T$: l'intensità precedente nel punto centrale $\mathbf{p}$ non viene considerata dato che verrà modificata dalla proiezione.



istogramma finale $\mathcal{H}(i)$ con intensità i^* a distanza massima

Figura 3.17: istogramma finale $\mathcal{H}(i)$ costruito a partire da $\mathcal{H}_R(i)$ e $\mathcal{H}_T(i)$.

Successivamente si utilizza l'istogramma $\mathcal{H}(i)$ per trovare un'intensità i^* non utilizzata nelle finestre ($\mathcal{H}(i^*) = 0$), tale che sia a distanza massima dalle intensità vicine utilizzate nell'istogramma:

$$i^* = \arg\max_i \big\{\text{dist}(i)\big\} \tag{3.12}$$

$$\text{dist}(i) = \big\{\min\big\{|i - i_s|,\ |i - i_d|\big\},\ i_s \in [0, i[\colon \mathcal{H}(i_s) > 0,\ i_d \in]i, L[\colon \mathcal{H}(i_d) > 0\big\} \tag{3.13}$$

L'intensità i^* verrà utilizzata per colorare la proiezione in $I(\mathbf{p})$: anche in questo caso c'è side-effect e l'istogramma dei vicini di $\mathbf{p}$ subirà modifiche.

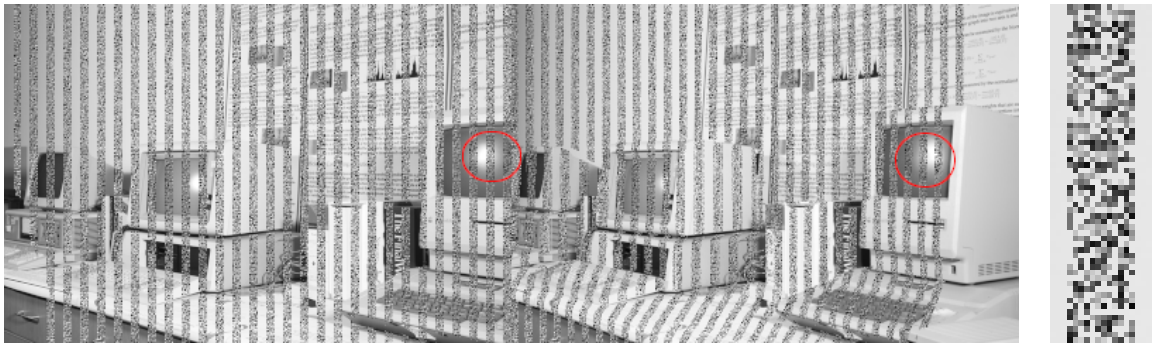


Figura 3.18: pattern cromatico a distanza massima e pattern spaziale a strisce, con vista reference (sinistra) e target (destra): si può notare che il pattern è più efficace nelle superfici non lambertiane (cerchio rosso) rispetto al pattern negativo.

Pattern cromatico casuale Il colore $I(\mathbf{p})$ è scelto casualmente tra le tonalità disponibili: possiamo associare $I(\mathbf{p})$ ad una variabile aleatoria con distribuzione uniforme.

$$I(\mathbf{p}) \sim \mathcal{U}(0, L-1) \tag{3.14}$$

Il vantaggio di questo pattern è che la probabilità che un punto proiettato sia colorato in modo ambiguo in tutte le iterazioni diminuisce con il numero di iterazioni (Eq. 3.15, Eq. 3.16), essendo $I(\mathbf{p})$ indipendente nelle iterazioni e nel side-effect.

$$P(\text{“}I(\mathbf{p}) \text{ è localmente ambiguo”}) = K \in [0, 1] \tag{3.15}$$

$$\begin{aligned} &P(\text{“}I(\mathbf{p}) \text{ è localmente ambiguo in tutte } n \text{ iterazioni”}) = \\ &\bigcap_{i=1}^{n} P(\text{“}I(\mathbf{p}) \text{ è localmente ambiguo”}) = K^n \leq K,\ n = 2, \dots \end{aligned} \tag{3.16}$$

Possiamo stimare K utilizzando la 3.13: supponiamo che $I(\mathbf{p})$ sia localmente ambiguo se ho una distanza ravvicinata alle intensità dell'istogramma.

$$K \propto \frac{1}{\max\{1, \text{dist}(I(\mathbf{p}))\}} \tag{3.17}$$

Effettuando test empirici (che tengono conto del side-effect di proiezione) si dimostra che $K \approx 0.35$: nella maggior parte delle iterazioni il colore scelto non sarà ambiguo, essendo un valore al di sotto del 50%.

Figura 3.19: pattern cromatico casuale e pattern spaziale casuale con 2 iterazioni (vista da camera reference): si può notare che uno stesso punto ha colori diversi nelle iterazioni.

Pattern cromatico census A differenza dei pattern cromatici visti precedentemente, questo pattern tiene conto esplicitamente della trasformata di census e della distanza di Hamming: date le stringhe di bit $C_R(\mathbf{p}_R)$ e $C_T(\mathbf{p}_T)$, lo scopo del pattern è ridurre la distanza di Hamming a zero.

Per ridurre la distanza di Hamming, bisogna modificare esplicitamente le stringhe di bit:

- si possono modificare le intensità $I_R(\mathbf{p}_{R1}), \ldots, I_T(\mathbf{p}_{Tn})$ delle finestre di supporto (metodo "census");
- si può modificare solo le intensità $I_R(\mathbf{p}_R)$ e $I_T(\mathbf{p}_T)$ (metodo "censusV2").

Il metodo "census" sceglie se negare il bit i-esimo in $C_R(\mathbf{p}_R)$ o $C_T(\mathbf{p}_T)$ minimizzando il costo relativo al cambiamento d'intensità:

$$\begin{cases} C'_R(\mathbf{p}_R)_i = \neg C_R(\mathbf{p}_R)_i & \text{se } |I_R(\mathbf{p}_{Ri}) - I_R(\mathbf{p}_R)| \leq |I_T(\mathbf{p}_{Ti}) - I_T(\mathbf{p}_T)| \\ C'_T(\mathbf{p}_T)_i = \neg C_T\mathbf{p}_T)_i & \text{se } |I_R(\mathbf{p}_{Ri}) - I_R(\mathbf{p}_R)| > |I_T(\mathbf{p}_{Ti}) - I_T(\mathbf{p}_T)| \end{cases} \tag{3.18}$$

Per modificare il bit i-esimo occorre modificare l'intensità del pixel della finestra di supporto associato al bit:

$$C'(\mathbf{p})_i = \neg C(\mathbf{p})_i \rightarrow \begin{cases} I'(\mathbf{p}_i) = I(\mathbf{p}) & \text{se } I(\mathbf{p}) > I(\mathbf{p}_i) \\ I'(\mathbf{p}_i) = I(\mathbf{p}) - 1 & \text{se } I(\mathbf{p}) \leq I(\mathbf{p}_i) \end{cases} \tag{3.19}$$

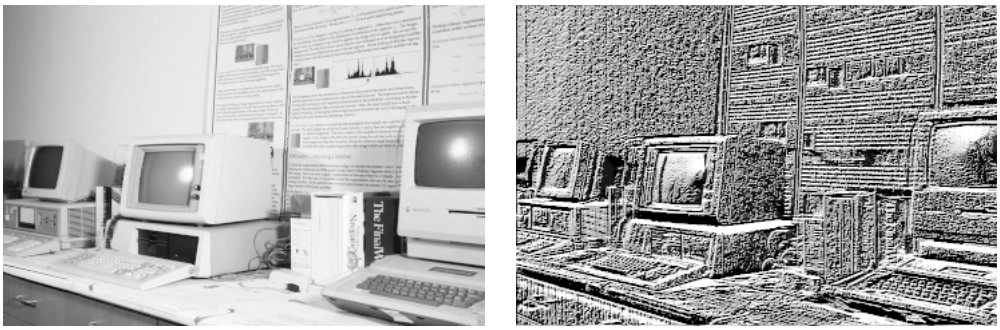


Figura 3.20: pattern cromatico "census" e pattern spaziale a scansioni: il pattern è poco visibile perché minimizza i cambi d'intensità, però risulta comunque efficace nel dominio della trasformata di census.

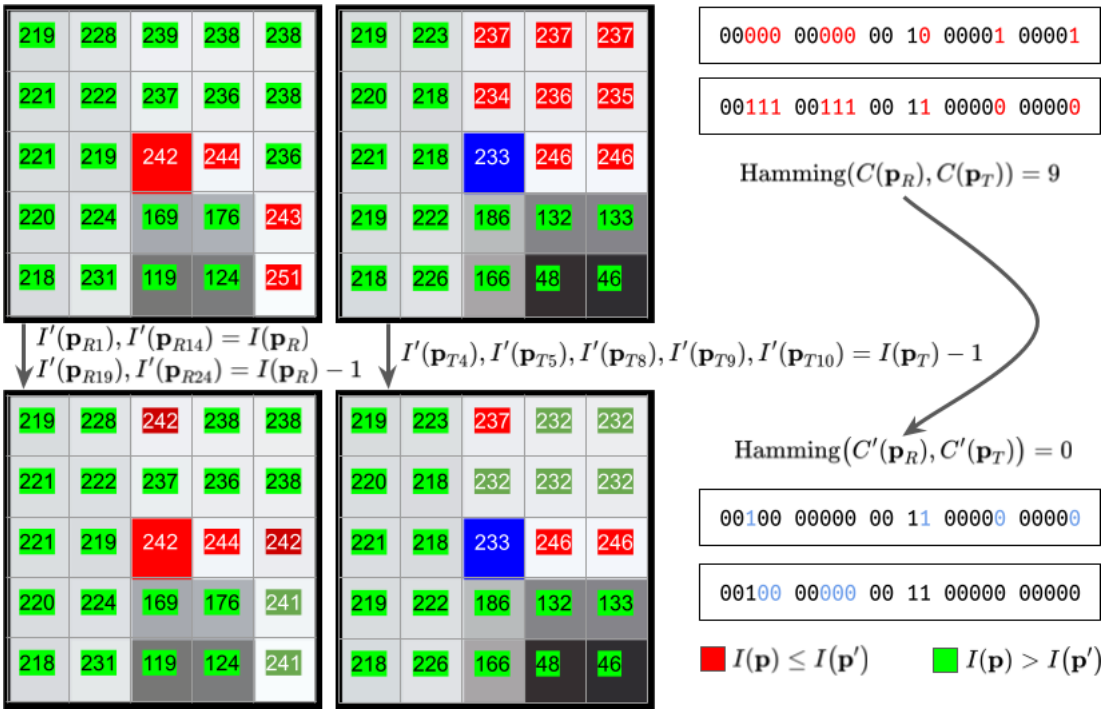


Figura 3.21: esempio di applicazione del pattern cromatico census: il bit viene modificato o in $C_R(\mathbf{p}_R)$ o in $C_T(\mathbf{p}_T)$ ( ).

Il metodo "censusV2" invece cerca di negare il bit i-esimo in $C_R(\mathbf{p}_R)$ o $C_T(\mathbf{p}_T)$ agendo solamente su $I_R(\mathbf{p}_R)$ e $I_T(\mathbf{p}_T)$: per evitare effetti collaterali ai bit che hanno già soddisfatto il vincolo imposto dalla distanza di Hamming, occorre impostare dei limiti al nuovo valore di $I'(\mathbf{p})$:

$$\begin{gathered} I'_R(\mathbf{p_R}) \in [i_{Rs}, i_{Rd}] \\ i_{Rs} \in [0, I_R(\mathbf{p}_R)[: i_{Rs} = \max\{I_R(\mathbf{p}_{R1}), \ldots, I_R(\mathbf{p}_{Rn})\} \\ i_{Rd} \in]I_R(\mathbf{p}_R), L[: i_{Rd} = \min\{I_R(\mathbf{p}_{R1}), \ldots, I_R(\mathbf{p}_{Rn})\} \\ \mathbf{p}_{R1}, \ldots, \mathbf{p}_{Rn} \in \mathcal{N}'_{\mathbf{p}_R} \end{gathered} \tag{3.20}$$

$$\begin{gathered} I'_T(\mathbf{p_T}) \in [i_{Ts}, i_{Td}] \\ i_{Ts} \in [0, I_T(\mathbf{p}_T)[: i_{Ts} = \max\{I_T(\mathbf{p}_{T1}), \ldots, I_T(\mathbf{p}_{Tn})\} \\ i_{Td} \in]I_T(\mathbf{p}_T), L[: i_{Td} = \min\{I_T(\mathbf{p}_{T1}), \ldots, I_T(\mathbf{p}_{Tn})\} \\ \mathbf{p}_{T1}, \ldots, \mathbf{p}_{Tn} \in \mathcal{N}'_{\mathbf{p}_T} \end{gathered} \tag{3.21}$$

Con $\mathcal{N}'_{\mathbf{p}_R}$ e $\mathcal{N}'_{\mathbf{p}_T}$ le finestre di supporto che considerano solo i pixel che hanno già soddisfatto la distanza di Hamming. I limiti imposti rendono il pattern non ottimo, in quanto non è sempre possibile raggiungere una soluzione con distanza di Hamming pari a zero.

Successivamente bisogna calcolare le nuove intensità $I'_R(\mathbf{p}_R)$ e $I'_T(\mathbf{p}_T)$, prima minimizzando la distanza di Hamming ($\text{Hamm}(C'_R(\mathbf{p}_R), C'_T(\mathbf{p}_T))$ e successivamente minimizzando le differenze con le intensità originali ($|I'_R(\mathbf{p}_R) - I_R(\mathbf{p}_R)|$ e $|I'_T(\mathbf{p}_T) - I_T(\mathbf{p}_T)|$).

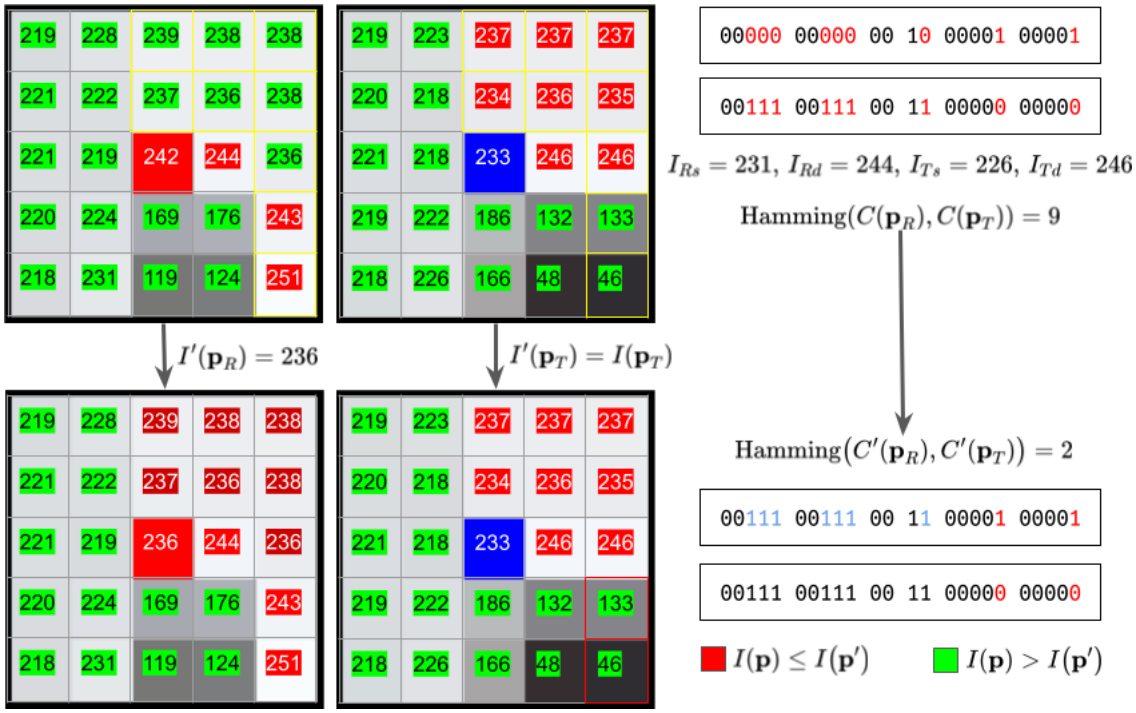


Figura 3.22: esempio di applicazione del pattern cromatico "censusV2" con i limiti $i_{Rs}, i_{Rd}, i_{Ts}, i_{Td}$ e le nuove intensità $I'_R(\mathbf{p_R})$ e $I'_T(\mathbf{p_T})$: in questo caso non si raggiunge l'ottimalità per via dei limiti.

I vincoli in "censusV2" potrebbero essere rilassati, tuttavia al costo di una ricerca più dispendiosa.

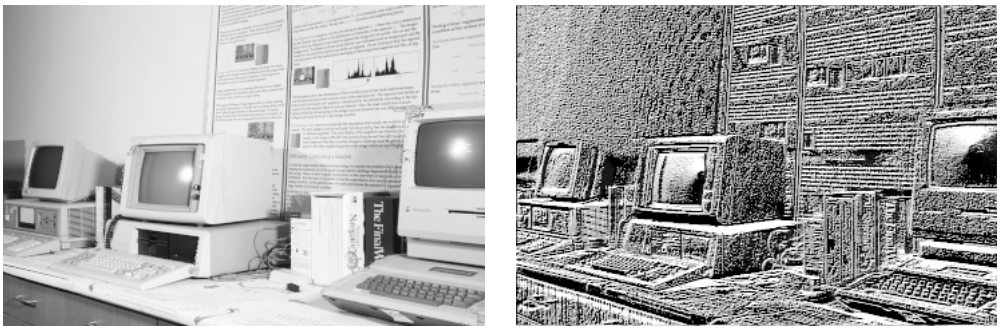


Figura 3.23: pattern cromatico "censusv2" e pattern spaziale a scansioni: il pattern è poco visibile perché minimizza i cambi d'intensità, però risulta comunque efficace nel dominio della trasformata di census, anche se leggermente meno rispetto al metodo "census".

Refinement globale Tutti i pattern cromatici visti precedentemente lavorano a livello locale e hanno lo svantaggio di generare un side-effect, più o meno evidente: è possibile effettuare una raffinazione globale, verificando che i punti affidabili siano proiettati correttamente.

Iperparametri	Tipo	Default	Descrizione
`colorMethod`	{"rnd", "negative", "mdist", "census", "censusv2"}	"rnd"	seleziona la tipologia di pattern cromatico
`useRGB`	booleano	falso	per applicare un pattern a colori o a scala di grigi
`wsizeAgg`	$1, 3, 5, \ldots$	5	dimensione della finestra di supporto $\mathcal{N}$ ("census" forza il valore a 5 e "censusV2" forza il valore a 1)

Tabella 3.6: iperparametri principali del pattern cromatico.

Il pattern cromatico è stato ottimizzato mediante Cython per velocizzare il blocco: tutte le modalità sono incluse nell'ottimizzazione.

Rumore Gaussiano

Inizialmente il rumore Gaussiano è stato introdotto per simulare un filtro temporale: lo scopo era quello di compensare il rumore generato intrinsecamente dalle camere reali.

$$I(\mathbf{m}) = \tilde{I}(\mathbf{m}) + n(\mathbf{m}) \quad n(\mathbf{m}) \sim \mathcal{N}(0, \sigma^2) \tag{3.22}$$

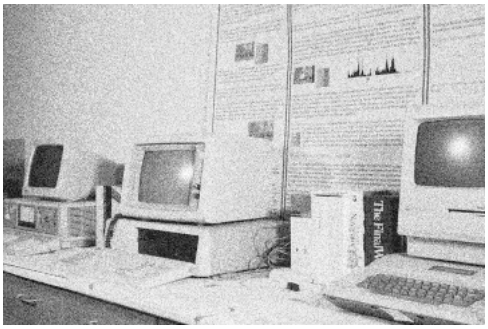
L'intensità reale I nel punto generico $\mathbf{m} = [u\ v]^T$ può essere modellata come la somma dell'intensità ideale $\tilde{I}(\mathbf{m})$ con il rumore Gaussiano $n(\mathbf{m}) \sim \mathcal{N}(0, \sigma^2)$ avente media $\mu = 0$ e varianza σ^2.

$$\mathbb{E}[I(\mathbf{m})] = \mathbb{E}[\tilde{I}(\mathbf{m})] + \mathbb{E}[n(\mathbf{m})] = \tilde{I}(\mathbf{m}) \tag{3.23}$$

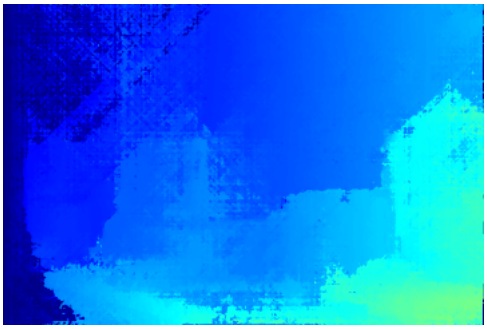
Se scattiamo una serie di immagini della stessa scena, possiamo ottenere una media temporale che rimuove l'errore, tuttavia la scena deve rimanere statica:

$$\mathbb{E}[I(\mathbf{m})] \approx \frac{1}{N} \sum_{t=1}^{N} I_t(\mathbf{m}) \tag{3.24}$$

Con N sufficientemente grande e $I_t(\mathbf{m})$ l'intensità registrata all'istante t nel punto $\mathbf{m}$. Tuttavia la proiezione virtuale lavora nel dominio delle iterazioni e non del tempo: l'idea era quella di applicare un rumore Gaussiano artificiale $n'(\mathbf{m}) \sim \mathcal{N}(0, \sigma^2)$ per simulare la serie di immagini scattate nel tempo, purtroppo però non si conosce realmente il valore di σ^2 per ogni punto.



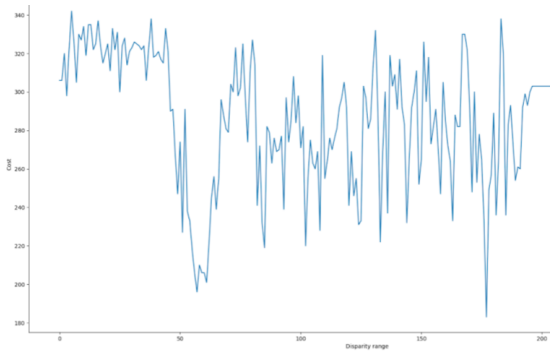
con rumore



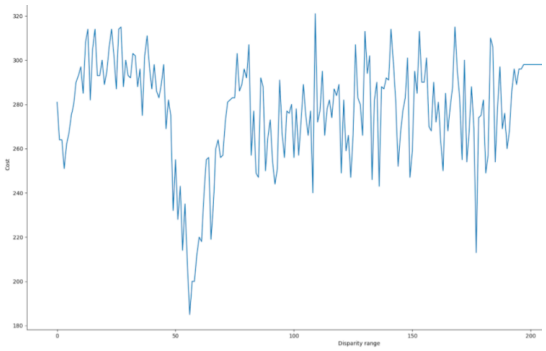
mappa di disparità

Figura 3.24: esempio di effetto del rumore Gaussiano: si può notare una grande perdita di dettaglio (il rumore è molto forte), anche se recupera nella superficie non lambertiana nel monitor del computer più a destra.

Il lavoro svolto sul rumore ha però permesso di dimostrare empiricamente un fenomeno di regolarizzazione che avviene usando algoritmi di matching globali o semiglobali: applicando il rumore Gaussiano su una superficie eventuali salti di disparità presenti diventano meno evidenti.



senza rumore



con rumore

Figura 3.25: esempio di effetto di regolarizzazione causata dal rumore Gaussiano: si può notare un cambiamento nel minimo.

Gli effetti del rumore Gaussiano sul matching sono sia positivi sia negativi:

- permette di regolarizzare l'immagine: l'effetto è positivo solo se l'intorno è alla stessa disparità (non lo sappiamo a priori);
- perdita di dettaglio se eccessivo: si ha un effetto negativo se l'effetto di regolarizzazione introdotta dagli algoritmi globali è troppo forte.

Questi risultati assomigliano agli effetti di un filtro Gaussiano su un'immagine [27]: aumentando σ aumenta la "scala" d'interesse, cioè i piccoli dettagli dell'immagine svaniscono, lasciando solo le strutture maggiori.

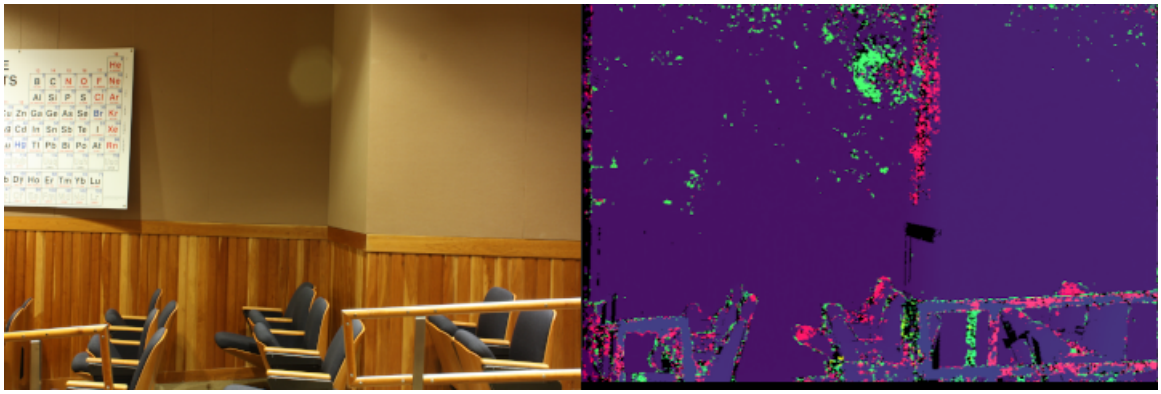


Figura 3.26: effetti del rumore Gaussiano[11]: il rumore ha effetti positivi (verde) sulla parte parallela del muro, tuttavia ha anche effetti negativi (rosso) sull'angolo del muro al centro (l'algoritmo globale regolarizza "appiattendo" il muro).

Un indicatore che distingue le zone uniformi dalle zone con dettagli fini potrebbe essere utilizzato per aggiungere il rumore solo dove veramente (rumore Gaussiano adattivo).

Per verificare se il rumore Gaussiano è necessario per tutte le iterazioni o conviene ridurlo, è stata creta una funzione modulatrice $G_\gamma(i)$, in grado di mantenere o diminuire il rumore a seconda della iterazione corrente i e del valore di threshold γ:

$$G_\gamma(i) = \begin{cases} 1 & \text{se } i \leq \gamma \\ \frac{\gamma}{i+1} & \text{se } i > \gamma \end{cases} \tag{3.25}$$

[11] Fonte immagine: Classroom - Middlebury2014 [5].

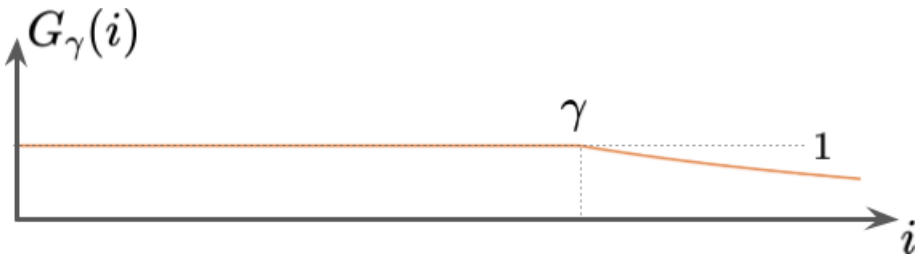


Figura 3.27: funzione modulatrice $G_\gamma(i)$.

Iperparametri	Tipo	Default	Descrizione
`std_dev`	$[0.0, 1.0]$	0.0	imposta la deviazione standard σ
`std_dev_th`	$[0.0, 1.0]$	0.0	imposta il threshold γ sotto forma di percentuale sul numero di iterazioni

Tabella 3.7: iperparametri principali del rumore Gaussiano.

Il rumore Gaussiano viene generato velocemente tramite la libreria Cython e la trasformazione di Box-Muller [28].

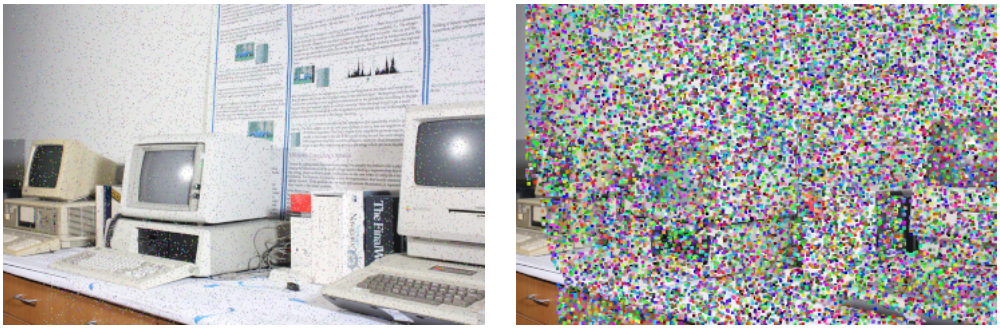
Dimensione dei punti

Di norma i punti proiettati sui piani d'immagine occupano solo un pixel (se non consideriamo l'interpolazione): la proiezione $\mathbf{p}$ ha una dimensione 1×1.

Se però ipotizziamo che le disparità attorno al punto $\mathbf{p}$ siano uguali allora stiamo assumendo che il punto si trovi in una superficie fronto-parallela.

$$\forall \mathbf{p}_k \in \mathcal{N}_\mathbf{p} : I(\mathbf{p}_k) \approx I(\mathbf{p}) \tag{3.26}$$

Di conseguenza ora conosciamo altri punti da proiettare su schermo, perché conosciamo la disparità del punto affidabile $\mathbf{p}$: possiamo applicare il pattern cromatico non solo ad un solo pixel, ma ad una finestra $\mathcal{N}_\mathbf{p}$ (e.g., 3×3, 5×5).



1×1 5×5

Figura 3.28: esempio di dimensione dei punti 1×1 a confronto con una finestra 5×5.

Anche la dimensione dei punti ha una funzione modulatrice $\bar{G}_\gamma(i)$ discreta per ridurre gradualmente la dimensione della finestra (i.e., 7×7, 5×5, 3×3, fino a 1×1):

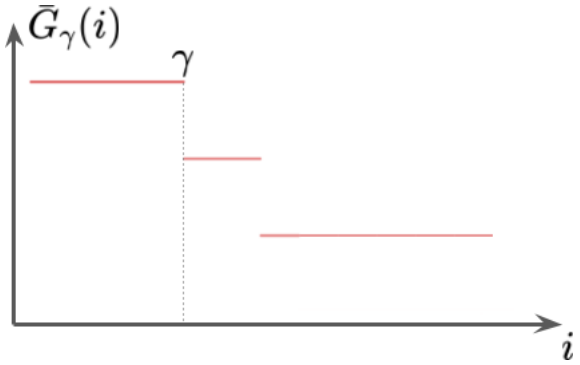


Figura 3.29: funzione modulatrice $\bar{G}_\gamma(i)$.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`wsize`	$1, 3, 5, \ldots$	1	imposta la dimensione dei punti
`wsize_th`	$[0.0, 1.0]$	0.0	imposta il threshold γ sotto forma di percentuale sul numero di iterazioni

Tabella 3.8: iperparametri principali della dimensione dei punti.

L'implementazione su Cython unisce pattern spaziale, pattern cromatico e dimensione dei punti in un'unica funzione ottimizzata.

3.2.6 Aggregazione del DSI

Durante le iterazioni i volumi di costo parziali $\mathrm{DSI}_i(\mathbf{m}, d)$ restituiti dall'algoritmo di matching alla iterazione i-esima vengono aggregati in un volume di costo unico $\mathrm{IDSI}(\mathbf{m}, d)$:

$$\mathrm{IDSI}(\mathbf{m}, d) = \overline{\mathrm{DSI}}(\mathbf{m}, d) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{DSI}_i(\mathbf{m}, d) \tag{3.27}$$

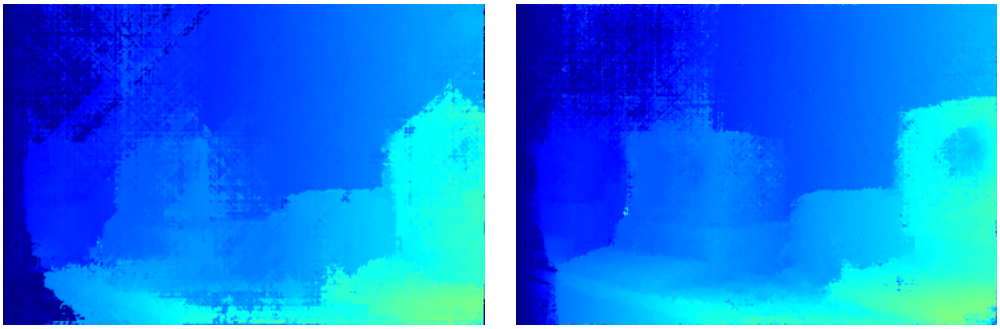
Lo stadio di aggregazione e lo stadio di proiezione vengono coordinati dal coordinatore globale per eseguire il ciclo di proiezione: una volta finito il ciclo, il volume integrato è consegnato allo stadio di disparity computation per calcolare la mappa di disparità.

Rumore Gaussiano

Il rumore Gaussiano è stato applicato al volume di costo per verificare se l'effetto di regolarizzazione del rumore, applicato allo stadio di proiezione, sia causato dall'algoritmo semiglobale e non dallo stadio di pre-processing o matching.

$$\mathrm{IDSI}(\mathbf{m}, d) = \tilde{\mathrm{IDSI}}(\mathbf{m}, d) + n(\mathbf{m}, d) \sim \mathcal{N}(0, \sigma^2) \tag{3.28}$$

Dove $n(\mathbf{m}, d)$ è il rumore Gaussiano con media $\mu = 0$ e varianza σ^2 applicato al punto generico $\mathbf{m} = [u\ v]^T$ e disparità d.



proiezione ($\sigma = 0.10$) aggregazione ($\sigma = 0.33$)

Figura 3.30: confronto tra rumore inserito nello stadio di proiezione e rumore inserito nello stadio di aggregazione.

I risultati empirici dimostrano che il rumore Gaussiano applicato al volume di costo genera un effetto di regolarizzazione, precisando però che gli stadi intermedi di pre-processing e matching alterano il rumore applicato al blocco di proiezione.

Inoltre l'applicazione del rumore al livello di aggregazione sporca la proiezione dei punti affidabili: la tecnica è efficace nel regolarizzare l'immagine, ma è stata abbandonata.

Filtro di media

Si tratta di un concetto che abbiamo abbandonato per gli scarsi risultati ottenuti: l'idea di fondo è di filtrare il volume di costo parziale $\mathrm{DSI}_i(\mathbf{m}, d)$ con un filtro di media, con lo scopo di effettuare una pre-aggregazione dei costi alla stessa disparità:

$$\mathrm{DSI}'_i(\mathbf{m}, d) = \frac{1}{|\mathcal{N}_\mathbf{m}|} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{m}} \mathrm{DSI}_i(\mathbf{q}, d) \tag{3.29}$$

Iperparametri	Tipo	Default	Descrizione
`k_size`	$1, 3, 5, \ldots$	1	imposta la dimensione del filtro di media (e.g., 1×1, 3×3)
`dsi_std_dev`	$[0.0, 1.0]$	0.0	imposta la deviazione standard σ

Tabella 3.9: iperparametri principali della dimensione dei punti.

La funzione di rumore è stata ottimizzata come nel blocco di proiezione tramite Cython e la trasformazione di Box-Muller [28], mentre l'aggregazione è già molto veloce grazie alla libreria NumPy.

3.2.7 Guided Stereo Matching

Si tratta dello stadio che implementa la strategia proposta in [7]. Viene applicato al volume dei costi integrato $\mathrm{IDSI}(\mathbf{m}, d)$ e ha una funzione analoga allo stadio di proiezione: il volume dei costi viene modulato con una funzione modulatrice Gaussiana $G^+_\mathbf{p}(d)$ con lo scopo di iniettare la disparità conosciuta $d^*_g(\mathbf{p}) = u_{\mathbf{p}_R} - u_{\mathbf{p}_T}$ del punto affidabile $\mathbf{P}$:

$$\mathrm{IDSI}'(\mathbf{p}, d) = \mathrm{IDSI}(\mathbf{p}, d) \cdot G^+_\mathbf{p}(d) \tag{3.30}$$

$$G^+_\mathbf{p}(d) = k \cdot \left(1 - \exp\left(-\frac{(d - d^*_g(\mathbf{p}))^2}{2c^2}\right)\right) \tag{3.31}$$

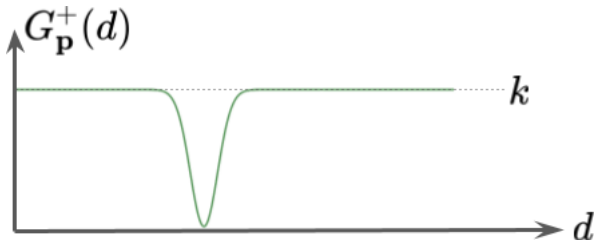
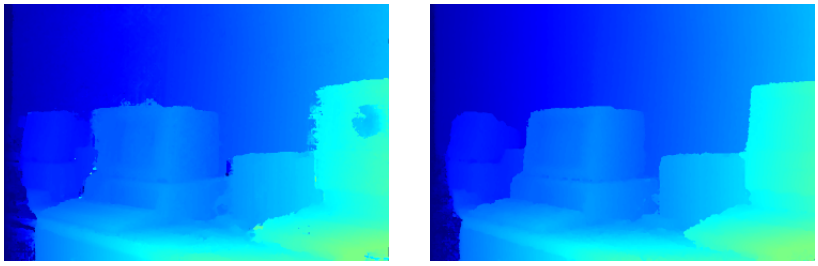


Figura 3.31: funzione modulatrice $G_{\mathbf{p}}^{+}(d)$.

Con k il guadagno della funzione modulatrice e c la deviazione della Gaussiana.



guided (RMSE = 4.22) guided+VPP (RMSE = 1.69)

Figura 3.32: confronto tra metodo Guided Stereo Matching e la combinazione Guided+VPP.

Iperparametri	Tipo	Default	Descrizione
`k`	reale	10 [7]	imposta il guadagno della funzione modulatrice
`c`	reale positivo	0.1 [7]	imposta la deviazione della funzione modulatrice

Tabella 3.10: iperparametri principali dello stadio guided.

Il blocco è stato implementato direttamente su Python, dato che viene eseguito solamente una volta nella pipeline.

3.2.8 Disparity computation

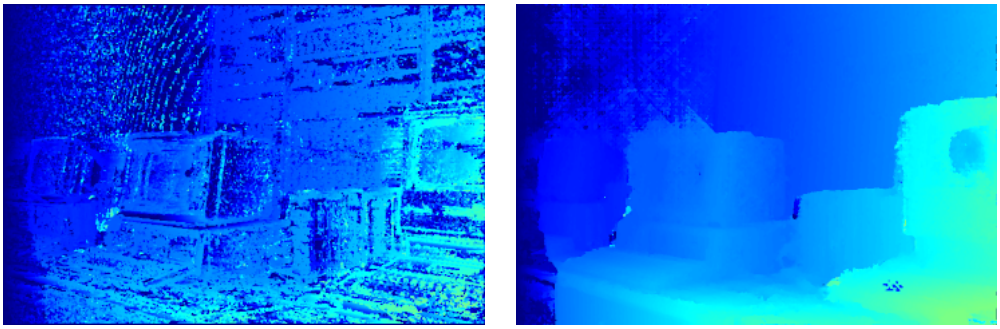
Lo stadio di computazione della disparità si occupa di trasformare il volume dei costi IDSI($\mathbf{m}, d$) in una mappa di disparità, secondo una strategia locale o globale: sono state implementate entrambe le strategie.

- La strategia locale è implementata con una semplice winner-takes-all e insieme all'aggregazione 5×5 della trasformata di census viene chiamata *Block Matching (BM)*;

$$\left\{ d^*(\mathbf{m}) \in [d_{\min}, d_{\max}] \middle| d^*(\mathbf{m}) = \arg\min_d \mathrm{IDSI}(\mathbf{m}, d) \right\} \tag{3.32}$$

- la strategia globale è SGM-8 ed è implementata tramite la libreria rSGM, come visto nella sottosezione 2.3.1: la penalità P_2 è stata rivisitata [29] (Eq. 3.33).

$$P_2' = \max\left\{P_2, -\alpha \cdot |I_R(\mathbf{m}) - I_R(\mathbf{m} - \mathbf{r})| + \gamma\right\} \tag{3.33}$$



BM (RMSE = 26) SGM (RMSE = 6.3)

Figura 3.33: confronto tra metodo Block Matching (BM) e Semi Global Matching (SGM).

Disparity refinement

Lo stadio di disparity refinement è stato integrato nello stadio di disparity computation. Il blocco integrato implementa:

- un filtro mediano disponibile nella versione veloce 3×3 offerto dalla libreria rSGM, oppure a kernel più grande ma non ottimizzato;
- una raffinazione della disparità vincitrice d^* nel punto $\mathbf{m}$ (*sub-pixel refinement*) tramite interpolazione equiangolare [30]:

$$R(\mathbf{m}, d) = \mathrm{IDSI}(\mathbf{m}, d^*(\mathbf{m}) + d)$$

$$\hat{d}^*(\mathbf{m}) = \begin{cases} \frac{1}{2}\frac{R(\mathbf{m},1) - R(\mathbf{m},-1)}{R(\mathbf{m},0) - R(\mathbf{m},-1)} & \text{se } R(\mathbf{m},1) < R(\mathbf{m},-1) \\ \frac{1}{2}\frac{R(\mathbf{m},1) - R(\mathbf{m},-1)}{R(\mathbf{m},0) - R(\mathbf{m},1)} & \text{altrimenti} \end{cases} \tag{3.34}$$

Lo sviluppatore della libreria rSGM ha ottimizzato il blocco, facendo largo uso delle istruzioni SIMD.

Iperparametri	Tipo	Default	Descrizione
`class`	{ "sgm", "bm" }	"sgm"	imposta la strategia di computazione
`median_k`	$3, 5, 7, \ldots$	3	imposta la dimensione del filtro mediano
`doRightMatch`	booleano	falso	calcola anche la mappa di disparità dalla vista di destra
`p1`	reale	11	penalizzazione P_1 nel costo SGM
`p2min`	reale	17	penalizzazione P_2 minima per il calcolo di P_2'
`alpha`	reale	0.5	fattore α per il calcolo di P_2'
`gamma`	reale	17	fattore γ per il calcolo di P_2'

Tabella 3.11: iperparametri principali dello stadio disparity computation.

3.2.9 Misure d'errore

Il blocco che misura l'errore dell'algoritmo stereo è collegato dal coordinatore globale allo stadio dataset per attingere la ground-truth $d_g^*(\mathbf{m})$ e al blocco disparity computation per attingere al risultato di matching $d^*(\mathbf{m})$.

Seguendo la letteratura [31], nel blocco sono state implementate varie misure d'errore:

- *Mean Square Error (MSE)*:

$$\text{MSE} = \frac{1}{|\mathbf{I}|} \sum_{\mathbf{m} \in \mathbf{I}} \left(d_g^*(\mathbf{m}) - d^*(\mathbf{m}) \right)^2 \tag{3.35}$$

- *Root Mean Square Error (RMSE)*:

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{3.36}$$

- *Mean Relative Error (MRE)*:

$$\text{MRE} = \frac{1}{|\mathbf{I}|} \sum_{\mathbf{m} \in \mathbf{I}} \frac{\left| d_g^*(\mathbf{m}) - d^*(\mathbf{m}) \right|}{d_g^*(\mathbf{m})} \tag{3.37}$$

- *Bad Matching Pixel (BMP)* o *Error rate*:

$$\begin{aligned} \mathrm{BMP}_\delta &= \frac{1}{|\mathbf{I}|}\sum_{\mathbf{m}\in\mathbf{I}} \epsilon_\delta(\mathbf{m}) \\ \epsilon_\delta(\mathbf{m}) &= \begin{cases} 1 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| > \delta \\ 0 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| \leq \delta \end{cases} \end{aligned} \tag{3.38}$$

- *Bad Matching Pixel Relative Error (BMPRE)*:

$$\begin{aligned} \mathrm{BMPRE}_\delta &= \frac{1}{|\mathbf{I}|}\sum_{\mathbf{m}\in\mathbf{I}} \psi_\delta(\mathbf{m}) \\ \psi_\delta(\mathbf{m}) &= \begin{cases} \tau(\mathbf{m}) & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| > \delta \\ 0 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| \leq \delta \end{cases} \\ \tau(\mathbf{m}) &= \begin{cases} \dfrac{\left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right|}{d_g^*(\mathbf{m})} & \text{se } d_g^*(\mathbf{m}) > 0 \\ 0 & \text{altrimenti} \end{cases} \end{aligned} \tag{3.39}$$

Nelle valutazioni quantitative verrà usato soprattutto RMSE e BMP con $\delta = 3$.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`th`	reale	3	imposta δ per gli errori BMP e BMPRE
`zero_check`	booleano	falso	vero se si vuole confrontare anche le disparità invalide ($d^* = 0$)
`leftocc_check`	booleano	falso	vero se si vogliono confrontare anche le occlusioni

Tabella 3.12: iperparametri principali dello stadio delle misure d'errore.

Tutte le misure d'errore sono state ottimizzate tramite Cython per ottenere una valutazione dei risultati più veloce.

3.3 Valutazione della strategia

Per verificare l'efficacia della strategia proposta sono stati effettuati una lunga serie di test quantitativi tramite un server fatto con pezzi di recupero: una CPU i5-3470 senza acceleratore gpGPU. Il task che ha richiesto più tempo è stata la ricerca degli iperparametri ottimali: si tratta di un'operazione con una crescita esponenziale al numero di iperparametri presenti. Dato il numero di iperparametri e le scarse prestazioni della macchina server sono state necessarie decine di ore per l'esecuzione dei vari test: i test hanno prodotto centinaia di megabyte di dati, analizzati successivamente con grafici e schemi.

3.3.1 Valutazione qualitativa

La valutazione qualitativa si focalizza nei risultati della ricerca dei migliori iperparametri: divideremo i risultati per bassa densità di punti affidabili ($\approx 3-5\%$) e per alta densità di punti affidabili ($\approx 95-100\%$), rispetto alle dimensioni del piano d'immagine.

Bassa densità di punti

- iterazioni: aumentando di molto il numero di iterazioni l'errore rimane invariato (comportamento asintotico), ciò può derivare dal fatto che abbiamo già estratto tutte le informazioni dai punti affidabili;
- densità dei punti affidabili: conviene utilizzare tutti i punti affidabili disponibili nella stessa iterazione;
- pattern spaziale: il miglior pattern spaziale è quello a scansioni data la sua peculiarità di riduzione delle occlusioni;
- pattern cromatico: non c'è molta differenza tra i pattern cromatici (eccetto "censusV2" che ha delle performance peggiori), tuttavia il metodo casuale e il metodo a distanza massima si sono comportati meglio;
- dimensione dei punti: l'assunzione che i punti siano su una superficie frontoparallela di piccole dimensioni (e.g., 5×5) è molto efficace per la riduzione dell'errore su basse densità;
- rumore Gaussiano: l'aggiunta di rumore Gaussiano è efficace nella riduzione dell'errore, perché probabilmente aiuta a regolarizzare la proiezione;

- funzioni modulatrici: le funzioni modulatrici per il rumore Gaussiano e per la dimensione dei punti non hanno un impatto significativo.

Alta densità di punti

- iterazioni: si ha un comportamento asintotico come nel caso a bassa densità, tuttavia sono richieste più iterazioni perché probabilmente il pattern cromatico fatica a rendere distinguibili i punti;
- Densità dei punti affidabili: conviene utilizzare tutti i punti affidabili disponibili nella stessa iterazione;
- pattern spaziale: il miglior pattern spaziale è quello a scansioni data la sua peculiarità di riduzione delle occlusioni;
- pattern cromatico: i pattern "census" hanno performance peggiori rispetto alle controparti, mentre il metodo casuale e il metodo a distanza massima si sono comportati meglio;
- dimensione dei punti: per densità molto alte (MiddEval3) è meglio impostare una dimensione 1×1 per evitare di perdere i dettagli più fini;
- rumore Gaussiano: l'aggiunta di rumore Gaussiano è efficace nella riduzione dell'errore, perché probabilmente aiuta a regolarizzare la proiezione;
- funzioni modulatrici: le funzioni modulatrici per il rumore Gaussiano e per la dimensione dei punti non hanno un impatto significativo.

3.3.2 Valutazione quantitativa

La valutazione quantitativa consiste in un ablation study dove si mettono a confronto SGM tradizionale, SGM con VPP, SGM con guided stereo matching e SGM con guided stereo matching unito a VPP.

Divideremo la valutazione in bassa densità di punti affidabili e alta densità di punti affidabili, come nella valutazione qualitativa.

Bassa densità di punti

Il test quantitativo a bassa densità simula correttamente un sensore attivo reale: dal sensore attivo si possono collezionare solo un numero limitato di punti per le ragioni discusse nei capitoli precedenti.

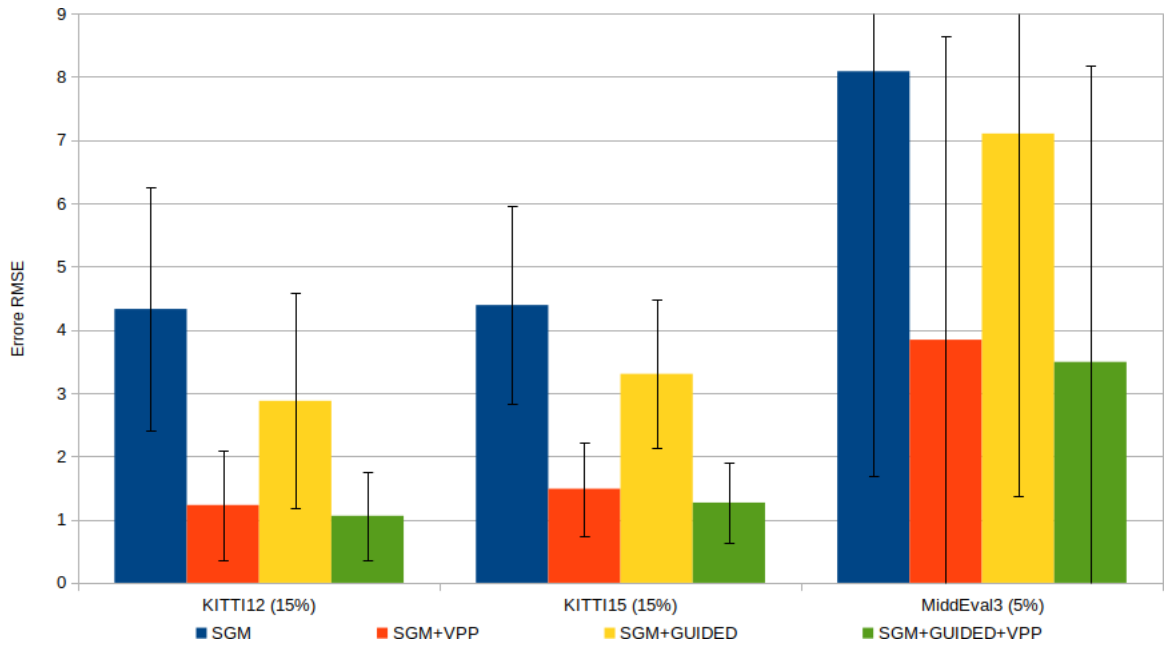


Figura 3.34: risultati qualitativi dell'ablation study, con densità dei punti affidabili al 5%: vengono confrontati i dataset KITTI12 [25], KITTI15 [26], MiddEval3 [5].

Si può notare dalla figura 3.34 che il dataset MiddEval3 possiede una grande varianza: ciò è dovuto alla presenza di scene variegate e studiate per stressare fortemente l'algoritmo di matching, mentre i dataset KITTI non presentano tale problema perché le scene urbane sono simili.

Variazione percentuale rispetto a SGM			
Dataset	**SGM+VPP**	**SGM+GUIDED**	**SGM+GUIDED+VPP**
KITTI12 (15%)	-72%	-34%	-74%
KITTI15 (15%)	-66%	-25%	-70%
MiddEval3 (5%)	-52%	-12%	-57%

Tabella 3.13: variazione percentuale dell'errore RMSE rispetto alla tecnica tradizionale SGM, con densità dei punti al 5%.

I risultati quantitativi a basse densità (Fig. 3.34, Tab. 3.13) mostrano l'efficacia del metodo VPP rispetto alla tecnica tradizionale SGM e alla tecnica Guided Stereo Matching [7], superando lo stato dell'arte: inoltre unendo le due tecniche si raggiungono risultati addirittura migliori.

Alta densità di punti

Il test quantitativo ad alta densità serve solamente per stressare le strategie di proiezione: se si conoscessero tutti i punti sul piano d'immagine, non ci sarebbe bisogno di utilizzare il sensore passivo, in quanto si hanno già tutte le informazioni a riguardo.

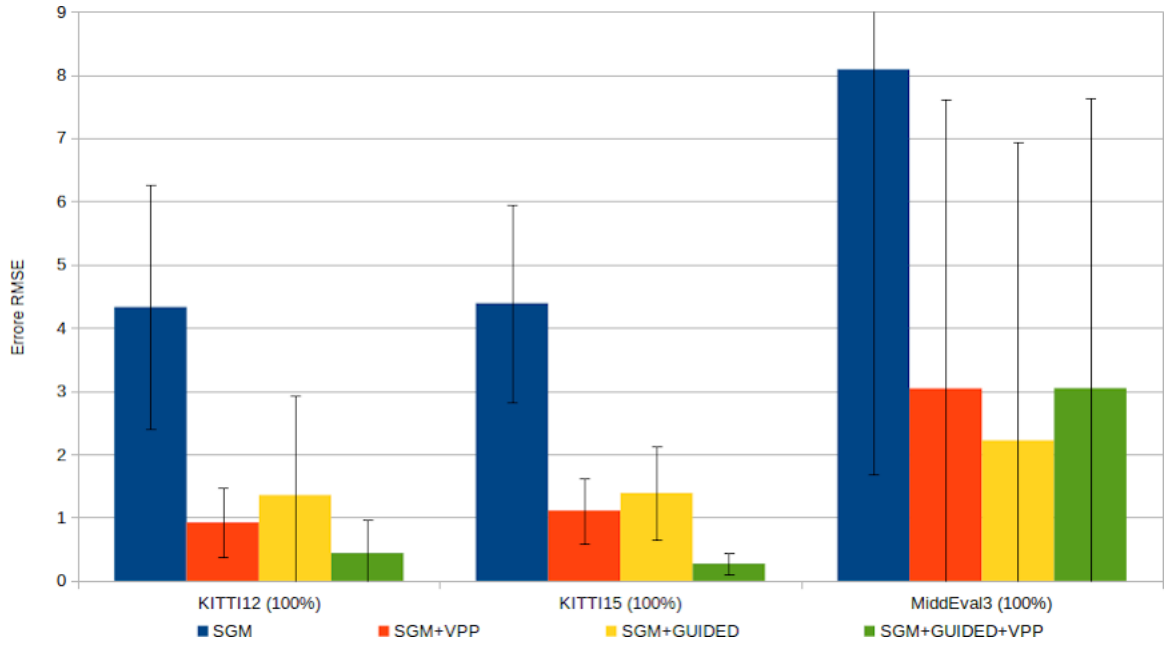


Figura 3.35: risultati qualitativi dell'ablation study, con densità dei punti affidabili al 100%: vengono confrontati i dataset KITTI12 [25], KITTI15 [26], MiddEval3 [5].

Anche se utilizziamo il 100% dei punti, non si ottiene una riduzione della varianza nel dataset MiddEval3 (Fig. 3.35): il dataset potrebbe essere utilizzato per migliorare la proiezione anche in condizioni difficili.

Variazione percentuale rispetto a SGM			
Dataset	**SGM+VPP**	**SGM+GUIDED**	**SGM+GUIDED+VPP**
KITTI12 (15%)	-79%	-69%	-90%
KITTI15 (15%)	-75%	-68%	-94%
MiddEval3 (5%)	-62%	-73%	-62%

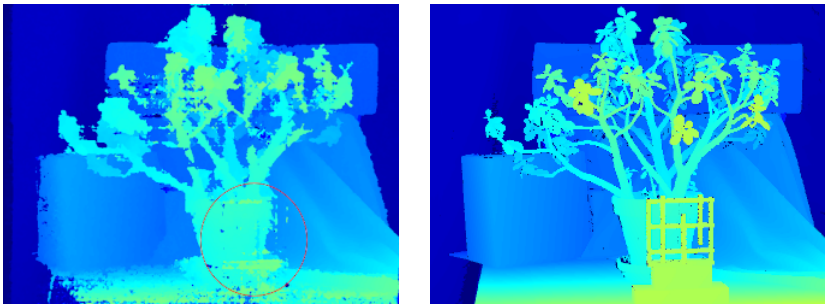
Tabella 3.14: variazione percentuale dell'errore RMSE rispetto alla tecnica tradizionale SGM, con densità dei punti al 100%.

I risultati quantitativi ad alta densità (Fig. 3.35, Tab. 3.14) mostrano l'efficacia del metodo VPP e del metodo VPP+GUIDED rispetto alla tecnica tradizionale SGM e alla tecnica Guided Stereo Matching nei dataset KITTI: in questi dataset i punti affidabili sono comunque sparsi, anche se si utilizza una densità al 100%, perché i dati sono raccolti tramite una tecnica diversa rispetto a [5].

Il metodo Guided Stereo Matching invece risulta migliore solamente nel caso di densità di punti affidabili veramente dense: non si tratta di un caso reale, però può essere preso come spunto per migliorare la tecnica VPP.

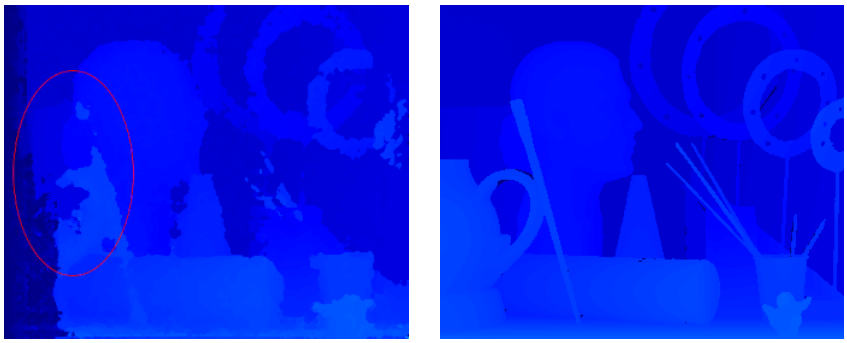
3.3.3 Fallimenti

Osservando i risultati della strategia nel dataset MiddEval3 si può notare una grande varianza ed un errore generalmente più alto rispetto ai dataset KITTI: in particolare l'aumento d'errore è derivato da un sottoinsieme limitato di scene.



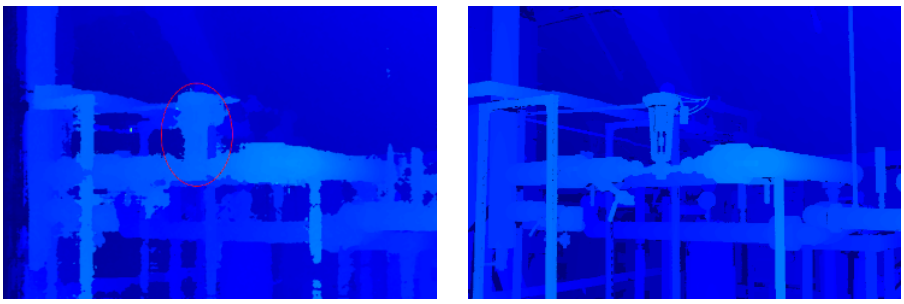
(Jadeplant - MiddEval3[12]- RMSE = 18)

Figura 3.36: fallimento 1: si può notare che l'oggetto a griglia in primo piano viene "nascosto" dai punti affidabili presi dal vaso.



(ArtL - MiddEval3[12] - RMSE = 9)

Figura 3.37: fallimento 2: si può notare una perdita dei dettagli fini e che il vaso non è formato correttamente.



(Pipes - MiddEval3[12] - RMSE = 7)

Figura 3.38: fallimento 3: si può notare perdite di dettagli fini.

Il problema maggiore che si nota dalle figure 3.36 3.37 e 3.38 è la perdita generalizzata di dettagli: ciò è dovuto principalmente da tre motivi.

1. Iperparametri di SGM: gli iperparametri di SGM decidono le penalità in caso di salti di disparità;
2. rumore Gaussiano: l'effetto di regolarizzazione del rumore Gaussiano nasconde i dettagli fini;
3. dimensione dei punti: l'assunzione di superfici fronto-parallele attorno ai punti affidabili fa perdere dettagli.

Si può intervenire cambiando algoritmo di matching oppure tarando tali iperparametri per limitare gli effetti negativi.

[12]Fonte immagine ground-truth: [5] (colorazione jet).

4. Misure di confidenza

Le misure di confidenza sono uno strumento utile per verificare effettivamente quali punti prodotti dall'algoritmo di matching siano effettivamente affidabili (confidenze positive) e quali siano da rivedere (confidenze negative).

Una misura di confidenza $c(\mathbf{m})$ attribuisce un valore reale ($c(\mathbf{m}) \in [0.0, 1.0]$) o binario ($c(\mathbf{m}) \in \{0, 1\}$) al punto $\mathbf{m}$: valori vicino a 1 indicano un'alta confidenza (un punto affidabile per le confidenze positive o un punto non affidabile per le confidenze negative), mentre valori prossimi allo zero indicano un'incertezza.

L'idea che svilupperemo in questo capitolo è l'analisi di fattibilità nell'utilizzo delle misure di confidenza per aiutare o sostituire eventuali sensori attivi come fonte di punti affidabili.

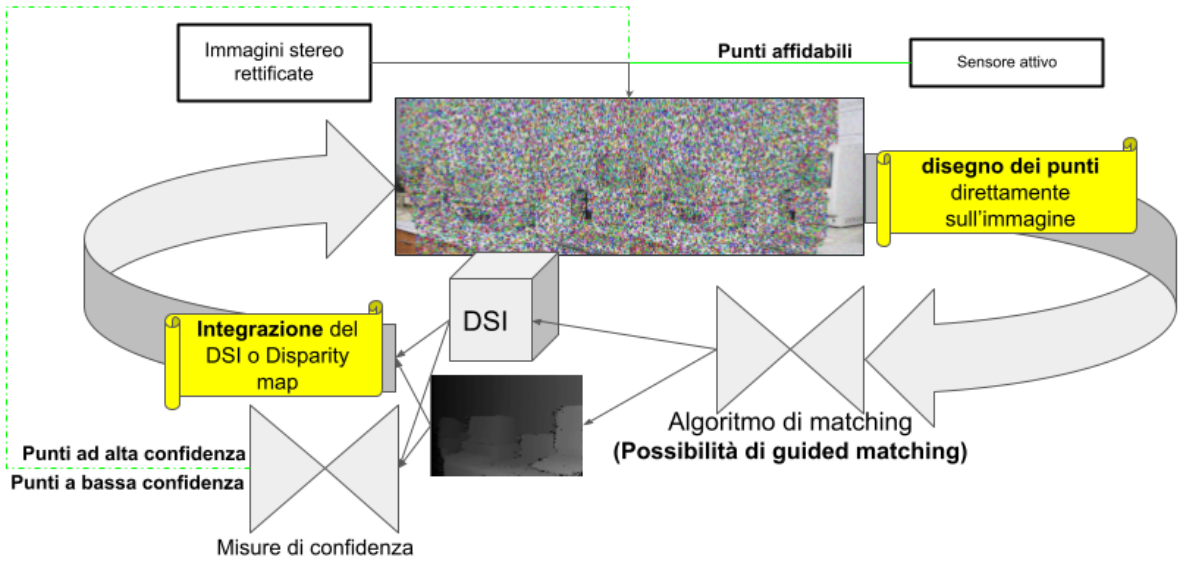


Figura 4.1: schema di funzionamento modificato della strategia VPP: i punti affidabili vengono estratti anche dalle misure di confidenza.

4.1 Pipeline

Nella pipeline modificata è stato introdotto uno stadio aggiuntivo per le misure di confidenza, ma anche uno stadio aggiuntivo di matching (linea tratteggiata in Fig. 4.2): è lo stadio collegato al blocco di confidenza per la generazione della mappa di disparità parziale.

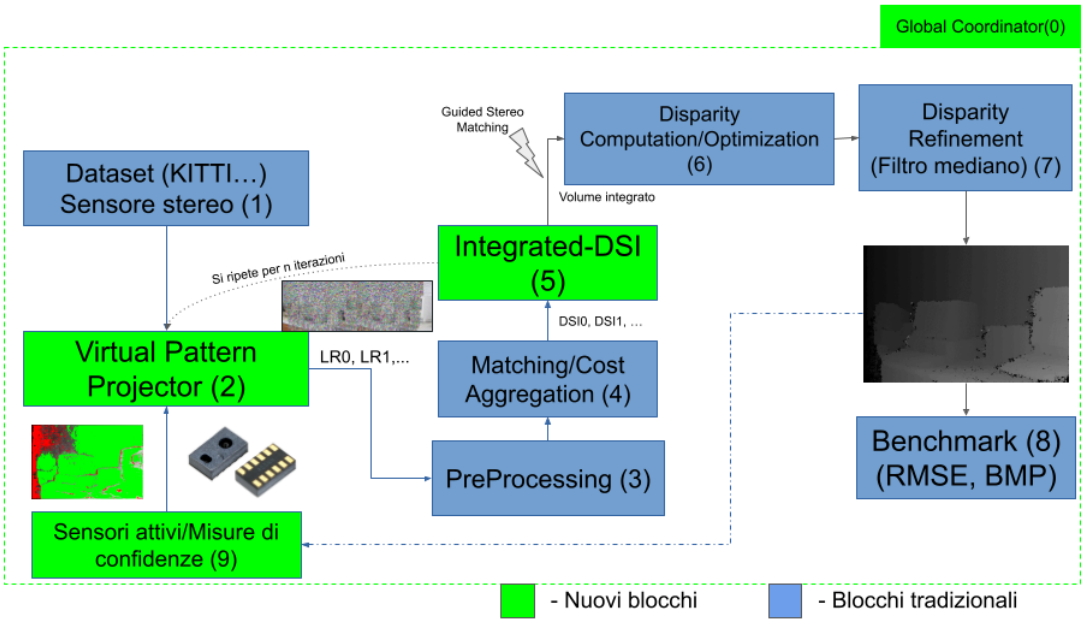


Figura 4.2: pipeline rivisitata con la modifica dei componenti per l'aggiunta delle misure di confidenza.

Inoltre sono necessarie modifiche principalmente allo stadio di proiezione e allo stadio di aggregazione per poter accettare le confidenze positive e le confidenze negative.

4.1.1 Coordinatore globale

Il coordinatore globale è stato modificato per accettare i nuovi blocchi di confidenza e matching aggiuntivo:

- il blocco aggiuntivo di matching viene eseguito ad ogni iterazione: il risultato è passato allo stadio di confidenza;
- il blocco di confidenza genera le confidenze positive e negative: sarà compito del coordinatore gestire i dati e consegnarli allo stadio di proiezione e allo stadio di aggregazione.

4.1.2 Proiezione virtuale

Lo stadio di proiezione ora supporta le confidenze positive e le confidenze negative: prima di essere applicate alla proiezione (attraverso qualche strategia) vengono aggregate nelle iterazioni tramite un meccanismo di consenso.

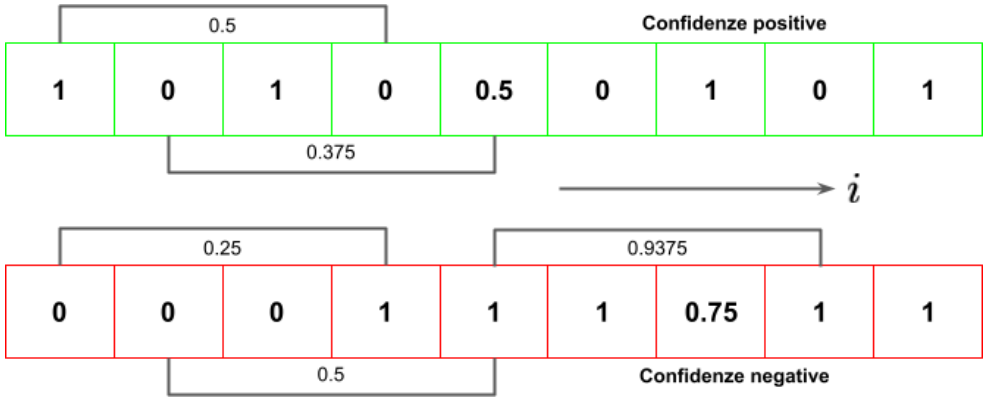


Figura 4.3: esempio del consenso con finestra a scorrimento (*sliding window*): calcolando la media si ottiene il valore del consenso. Se il valore supera una soglia (e.g., $T_h = 0.75$), allora la confidenza è accettata.

Indicheremo con $c^+(\mathbf{m}) \in [0,1]$ la mappa delle confidenze positive e con $c^-(\mathbf{m}) \in [0,1]$ la mappa delle confidenze negative, entrambe aggregate tramite il meccanismo del consenso. Il metodo adottato per proiettare differisce per confidenze positive e negative:

- le confidenze positive sono utilizzate per aumentare i punti affidabili:

$$c^+(\mathbf{p}^*) > T_h^+ \implies \mathbf{P}^* \text{ è affidabile} \tag{4.1}$$

  utilizzo la disparità $d^*(\mathbf{m})$ associata ai punti ad alta confidenza $c^+(\mathbf{m}) > T_h^+$ per proiettare $\mathbf{m}$ nelle due viste reference e target;

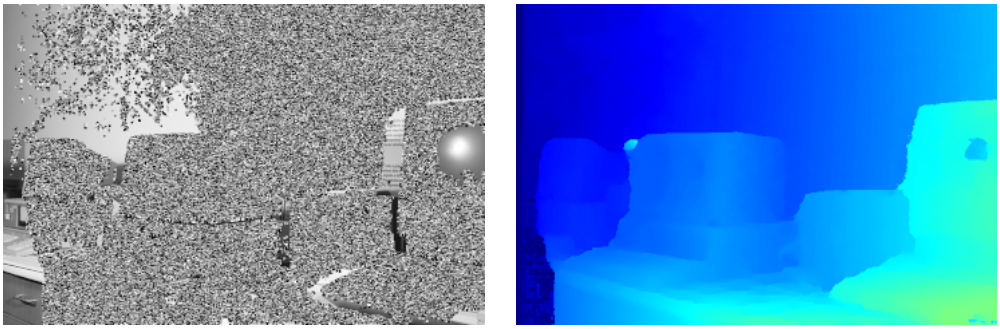


Figura 4.4: esempio di aggiunta di punti affidabili ottenuti da confidenza perfetta[1].

- le confidenze negative ($c^-(\mathbf{m}) > T_h^-$) indicano che una determina corrispondenza è sbagliata:

$$c^-(\mathbf{m}^*) > T_h^- \implies \mathbf{m}_R^* \nleftrightarrow \mathbf{m}_T^* \tag{4.2}$$

  si può utilizzare questa informazione come indicatore per il rumore Gaussiano adattivo: il rumore generato può portare ad una corrispondenza diversa, non necessariamente quella giusta.

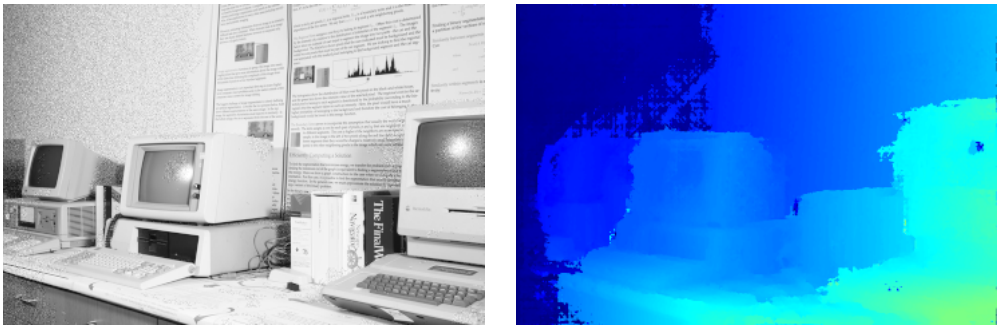


Figura 4.5: esempio di rumore Gaussiano adattivo usando la confidenza perfetta (oracolo[1]) come indicatore: è efficace nelle aree dove la zona da regolarizzare non è molto grande.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`confTh_plus`	$[0.0, 1.0]$	0.8	imposta la soglia T_h^+ per il meccanismo di consenso
`confTh_minus`	$[0.0, 1.0]$	0.8	imposta la soglia T_h^- per il meccanismo di consenso
`sws`	$1, 2, \ldots$	1	imposta la dimensione della finestra del consenso
`std_dev_mode`	booleano	falso	vero per attivare il rumore Gaussiano adattivo

Tabella 4.1: iperparametri aggiunti allo stadio di proiezione.

Le modifiche non hanno richiesto ottimizzazioni particolari: il meccanismo del consenso è stato implementato in Python e NumPy.

[1] Vedi l'approfondimento alla sottosottosezione 4.1.4.

4.1.3 Aggregazione del DSI

Lo stadio di aggregazione del DSI è stato modificato per accettare solo le confidenze negative, poiché la modifica del volume di costo a partire da punti affidabili è già stata implementata nel blocco Guided Stereo Matching. L'idea è quella di aumentare il costo nelle regioni di volume aventi una confidenza bassa:

$$c^-(\mathbf{m}^*) > T_h \implies \mathbf{m}_R^* \not\leftrightarrow \mathbf{m}_T^* \implies \text{IDSI}(\mathbf{m}^*, d^*(\mathbf{m}^*)) \text{ va aumentato} \tag{4.3}$$

Per provvedere ad aumentare il costo nella regione interessata è stata aggiunta una funzione modulatrice $G_{\mathbf{m}}^-(d)$ ispirata a quella utilizzata nel blocco Guided Stereo Matching:

$$\begin{aligned} \text{IDSI}'(\mathbf{m}, d) &= \text{IDSI}(\mathbf{m}, d) \cdot G_{\mathbf{m}}^-(d) \\ G_{\mathbf{m}}^-(d) &= 1 + k \cdot c^-(\mathbf{m}) \cdot \exp\left(-\frac{(d - d^*(\mathbf{m}))^2}{2c^2}\right) \end{aligned} \tag{4.4}$$

La funzione viene applicata in base alle confidenze negative: se $c^-(\mathbf{m})$ è bassa l'aumento del costo sarà irrisorio, invece se $c^-(\mathbf{m})$ è alta allora siamo più sicuri che si tratti di una corrispondenza sbagliata.

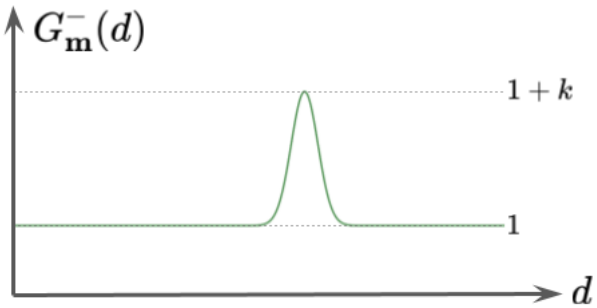


Figura 4.6: funzione modulatrice $G_{\mathbf{m}^*}^-(d)$, con $c^-(\mathbf{m}^*) = 1$.

Dato che la funzione modulatrice Gaussiana potrebbe aumentare anche il costo della corrispondenza giusta, si è deciso di "stringere" la funzione Gaussiana:

$$\begin{aligned} \text{IDSI}'(\mathbf{m}, d) &= \text{IDSI}(\mathbf{m}, d) \cdot \bar{G}_{\mathbf{m}}^-(d) \\ \bar{G}_{\mathbf{m}}^-(d) &= \lim_{c \to 0^+} G_{\mathbf{m}}^-(d) \end{aligned} \tag{4.5}$$

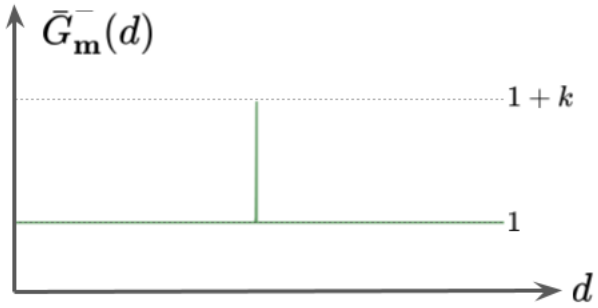


Figura 4.7: funzione modulatrice $\bar{G}_{\mathbf{m}}^{-}(d)$.

Assieme al rumore Gaussiano adattivo si tratta di uno dei metodi per utilizzare le confidenze negative: utilizzando assieme più metodi è possibile che si ottengano risultati migliori, come è successo nella valutazione quantitativa precedente.

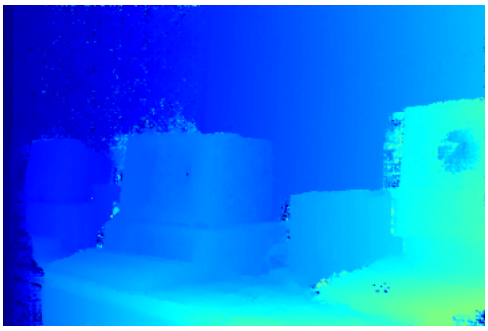


Figura 4.8: effetti della modulazione usando come indicatore la confidenza perfetta (oracolo): è efficace nelle zone a scarsa texture, di meno nelle regioni dove il rumore Gaussiano adattivo funziona bene.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`k`	reale	0.5	imposta il guadagno della funzione modulatrice
`c`	reale positivo	0.1	imposta la deviazione della funzione modulatrice (solo per $G_{\mathbf{m}}^{-}(d)$)
`notch_mode`	booleano	vero	vero per attivare la funzione modulatrice $\bar{G}_{\mathbf{m}}^{-}(d)$

Tabella 4.2: iperparametri aggiunti allo stadio di aggregazione.

Le modifiche non hanno richiesto ottimizzazioni particolari: le funzioni modulatrice sono state implementate in Python e NumPy.

4.1.4 Confidenza

Lo stadio di confidenza è il nuovo blocco che si occupa di gestire le misure di confidenza implementate e di trasformarle per renderle fruibili agli stadi di proiezione e aggregazione tramite il coordinatore globale.

Vedremo una serie di misure di confidenza allo stato dell'arte e nella sezione successiva la valutazione di ognuna.

Oracolo

Prima di vedere le misure di confidenza reali, approfondiremo la misura di confidenza "perfetta" (chiamata oracolo): utilizza la ground-truth $d_g^*(\mathbf{m})$ della vista di sinistra fornita dal dataset per generare le mappe di confidenza positive e negative.
Lo scopo è il tuning dei parametri di proiezione e aggregazione, senza che ci siano influenze da parte della misura di confidenza.

L'oracolo funziona secondo tre modalità distinte per verificare i limiti delle misure di confidenza:

1. modalità perfetta: le confidenze $c^+(\mathbf{m})$ e $c^-(\mathbf{m})$ non presentano errori di classificazione e la distribuzione delle mappe è uniforme su tutti i punti del piano d'immagine:

$$c^+(\mathbf{m}) = \begin{cases} 1 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| \leq \delta \\ 0 & \text{altrimenti} \end{cases}$$
$$c^-(\mathbf{m}) = \begin{cases} 1 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| > \delta \\ 0 & \text{altrimenti} \end{cases} \tag{4.6}$$

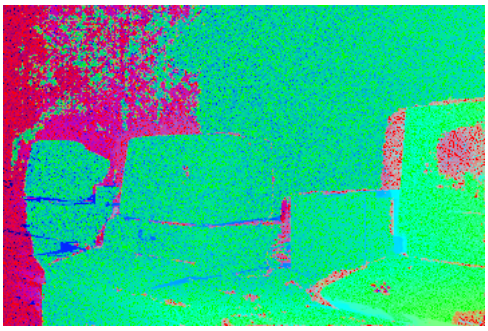


Figura 4.9: confidenza oracolo in modalità perfetta.

2. modalità a distribuzione sparsa: le mappe $c^+(\mathbf{m})$ e $c^-(\mathbf{m})$ vengono calcolate solo per un sottoinsieme $\tilde{\mathbf{I}}$ dei punti del piano d'immagine, aventi una distribuzione non uniforme:

$$c^+(\mathbf{m}) = \begin{cases} 1 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| \leq \delta \wedge \mathbf{m} \in \tilde{\mathbf{I}} \\ 0 & \text{altrimenti} \end{cases}$$
$$c^-(\mathbf{m}) = \begin{cases} 1 & \text{se } \left|d_g^*(\mathbf{m}) - d^*(\mathbf{m})\right| > \delta \wedge \mathbf{m} \in \tilde{\mathbf{I}} \\ 0 & \text{altrimenti} \end{cases} \tag{4.7}$$

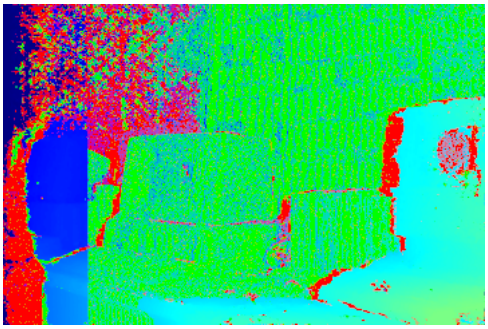


Figura 4.10: confidenza oracolo in modalità a distribuzione sparsa.

3. modalità a classificazione rumorosa: durante la generazione delle mappe $c^+(\mathbf{m})$ e $c^-(\mathbf{m})$ esiste una probabilità rispettivamente $K^+ > 0$ e $K^- > 0$ tale che un punto $\mathbf{m}$ sia classificato erroneamente.

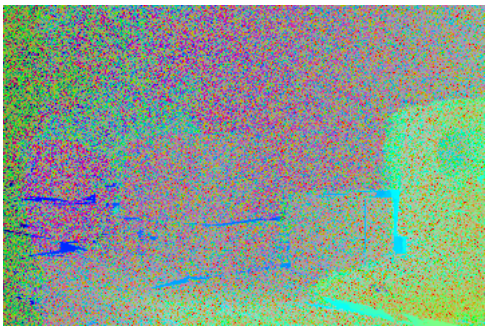


Figura 4.11: confidenza oracolo in modalità a classificazione rumorosa, con $K^+ = K^- = 0.33$.

Come si può notare dalle equazioni 4.6 e 4.7, la confidenza è binaria (i.e, $c^+(\mathbf{m})$, $c^-(\mathbf{m}) \in \{0, 1\}$).

Iperparametri	Tipo	Default	Descrizione
`oracle_mode`	$\{1, 2, 3\}$	1	imposta la modalità dell'oracolo
`bad_th`	reale positivo	3	imposta il threshold δ usato nella classificazione
`k_plus`	$[0.0, 1.0]$	0.01	imposta la probabilità di classificazione positiva K^+ errata
`k_minus`	$[0.0, 1.0]$	0.027	imposta la probabilità di classificazione negativa K^- errata
`density`	$[0.0, 1.0]$	1.0	imposta la densità della distribuzione uniforme

Tabella 4.3: iperparametri della confidenza oracolo.

Unsupervised Confidence Measures

La misura di confidenza *Unsupervised Confidence Measures (UCM)* racchiude al suo interno una serie di misure tradizionali, usate contemporaneamente per massimizzare le prestazioni [32]:

- *Uniqueness Constraint (UC)*: misura di confidenza binaria che codifica a bassa confidenza i punti che violano il vincolo di unicità.

$$\mathrm{UC}(\mathbf{m}) = \begin{cases} 0 & \text{se } \mathbf{m} \in Q \\ 1 & \text{altrimenti} \end{cases} \tag{4.8}$$
$$\mathrm{UC}^-(\mathbf{m}) = 1 - \mathrm{UC}(\mathbf{m}) \quad \mathrm{UC}^+(\mathbf{m}) = \mathrm{UC}(\mathbf{m})$$

  Dove $Q \subseteq \mathbf{I}_R$ è l'insieme dei punti reference che hanno la stessa corrispondenza nella vista target.

$$\mathbf{m}_R^* \in Q \iff \exists \mathbf{q}^* \in Q,\, \mathbf{m}_T^* \in \mathbf{I}_T :\ \mathbf{m}_R^* \leftrightarrow \mathbf{m}_T^* \wedge \mathbf{q}_R^* \leftrightarrow \mathbf{m}_T^* \tag{4.9}$$

- *Left Right Consistency (LRC)*: misura di confidenza binaria che codifica a bassa confidenza i punti corrispondenti che hanno disparità diversa nelle viste reference e target.

$$\mathrm{LRC}_\delta(\mathbf{m}) = \begin{cases} 0 & \text{se } |d^*(\mathbf{m}) - d^*_D(\mathbf{m}_T)| > \delta \\ 1 & \text{altrimenti} \end{cases} \tag{4.10}$$
$$\mathrm{LRC}^-_\delta(\mathbf{m}) = 1 - \mathrm{LRC}_\delta(\mathbf{m}) \quad \mathrm{LRC}^+_\delta(\mathbf{m}) = \mathrm{LRC}_\delta(\mathbf{m})$$

Dove $d^*(\mathbf{m})$ è la disparità dalla camera a sinistra (Eq. 2.20), $d^*_D(\mathbf{m}_T)$ è la disparità dalla camera a destra, infine $\mathbf{m}_R$ e $\mathbf{m}_T$ sono punti corrispondenti ($\mathbf{m}_R \leftrightarrow \mathbf{m}_T$) avendo come reference la vista di sinistra.

- *MEdian Deviation of disparity (MED)*: misura di confidenza binaria che codifica a bassa confidenza i punti che differiscono dalla disparità mediana in una finestra di supporto attorno al punto.

$$\mathrm{MED}_\delta(\mathbf{m}) = \begin{cases} 0 & \text{se } \left|d^*(\mathbf{m}) - \mathrm{median}\left[d^*(\mathbf{q}),\ \mathbf{q} \in \mathcal{N}_\mathbf{m}\right]\right| > \delta \\ 1 & \text{altrimenti} \end{cases} \tag{4.11}$$
$$\mathrm{MED}^-_\delta(\mathbf{m}) = 1 - \mathrm{MED}_\delta(\mathbf{m}) \quad \mathrm{MED}^+_\delta(\mathbf{m}) = \mathrm{MED}_\delta(\mathbf{m})$$

Dove $\mathrm{median}\left[\,\cdot\,\right]$ è l'operatore di filtro mediano, calcolato nella finestra di supporto $\mathcal{N}_\mathbf{m}$.

- *Distance to Left Border (DLB)*: misura di confidenza binaria che codifica a bassa confidenza i punti con distanza dal bordo di sinistra (Fig. 2.3) inferiore a $d_{\max}$.

$$\mathrm{DLB}(\mathbf{m}) = \begin{cases} 0 & \text{se } u_\mathbf{m} < d_{\max} \\ 1 & \text{altrimenti} \end{cases} \tag{4.12}$$
$$\mathrm{DLB}^-(\mathbf{m}) = 1 - \mathrm{DLB}(\mathbf{m}) \quad \mathrm{DLB}^+(\mathbf{m}) = \mathrm{DLB}(\mathbf{m})$$

Con $d_{\max}$ la disparità massima impostata nell'iperparametro `dmax` (Tab. 3.3).

- *Winner MargiN (WMN)*: misura di confidenza reale che codifica ad alta confidenza i punti che hanno un minimo nel volume dei costi evidente (normalizza la differenza tra minimi).

$$\mathrm{WMN}(\mathbf{m}) = \frac{\mathrm{DSI}(\mathbf{m}, d^*_2(\mathbf{m})) - \mathrm{DSI}(\mathbf{m}, d^*(\mathbf{m}))}{\sum_d \mathrm{DSI}(\mathbf{m}, d)} \tag{4.13}$$

Nella 4.13 $d^*_2(\mathbf{m})$ rappresenta il secondo minimo, cioè la disparità che ha un costo maggiore solamente a $d^*(\mathbf{m})$.

$$\begin{gathered}\left\{d_2^*(\mathbf{m}) \in [d_{\min}, d_{\max}] \middle| \forall d \neq d^*(\mathbf{m}),\ \mathrm{DSI}(\mathbf{m}, d_2^*(\mathbf{m})) \leq \mathrm{DSI}(\mathbf{m}, d)\right\} \\ \mathrm{DSI}(\mathbf{m}, d_2^*(\mathbf{m})) \geq \mathrm{DSI}(\mathbf{m}, d^*(\mathbf{m}))\end{gathered} \tag{4.14}$$

- *Average PeaK Ratio (APKR)*: misura di confidenza reale che codifica ad alta confidenza i punti che hanno un minimo nel volume dei costi evidente (effettua una media spaziale sul rapporto dei minimi).

$$\mathrm{APKR}(\mathbf{m}) = \frac{1}{|\mathcal{N}_{\mathbf{m}}|} \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{m}}} \frac{\mathrm{DSI}(\mathbf{q}, d_2^*(\mathbf{m}))}{\mathrm{DSI}(\mathbf{q}, d^*(\mathbf{m}))} \tag{4.15}$$

Le misure reali $c_{\mathcal{R}}(\mathbf{m})$ (i.e., APKR, WMN) vengono trasformate in misure binarie $c(\mathbf{m})$ calcolando un valore di soglia α_L e α_H sul percentile: per esempio se imposto $\alpha_L = 0.2$ e $\alpha_H = 0.8$, allora verranno classificate come negative $c^-(\mathbf{m})$ le prime confidenze entro il 20-esimo percentile e come positive $c^+(\mathbf{m})$ le ultime confidenze dopo l'80-esimo percentile, ordinando i valori reali di confidenza in ordine crescente.

$$c^+(\mathbf{m}) = \begin{cases} 1 & \text{se } c_{\mathcal{R}}(\mathbf{m}) \geq [c_{\mathcal{R}},\, \alpha_H]^{\text{th}} \\ 0 & \text{altrimenti} \end{cases} \qquad c^-(\mathbf{m}) = \begin{cases} 1 & \text{se } c_{\mathcal{R}}(\mathbf{m}) \leq [c_{\mathcal{R}},\, \alpha_L]^{\text{th}} \\ 0 & \text{altrimenti} \end{cases} \tag{4.16}$$

Nella 4.16 intendiamo con $[c_{\mathcal{R}},\, \alpha]^{\text{th}}$ l'operatore che calcola il α-esimo percentile sulla misura reale $c_{\mathcal{R}}$.

Le misure vengono aggregate tramite un semplice consenso totale, cioè per definire un punto ad alta confidenza tutte le misure interne devono definirlo tale.

$$\begin{aligned}\mathrm{UCM}^+(\mathbf{m}) &= \bigcap_{c_k^+ \in \mathcal{P}^+} c_k^+(\mathbf{m}),\ \mathcal{P}^+ = \{\text{UC, LRC, UC, DLB, MED, WMN, APKR}\} \\ \mathrm{UCM}^-(\mathbf{m}) &= \bigcap_{c_k^- \in \mathcal{P}^-} c_k^-(\mathbf{m}),\ \mathcal{P}^- = \{\text{UC, LRC, UC, WMN, APKR}\}\end{aligned} \tag{4.17}$$

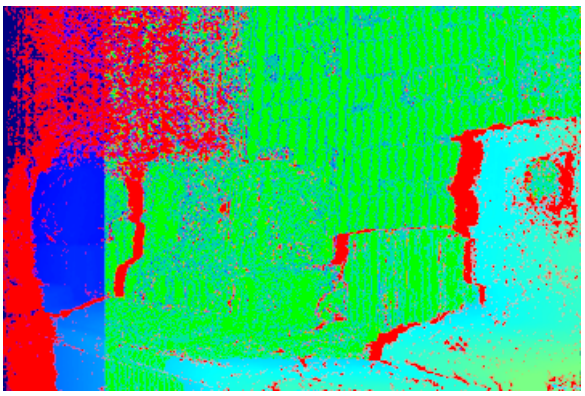


Figura 4.12: confidenza UCM, con $\alpha = 0.2$ e $\delta = 3$.

Anche in questa misura di confidenza è presente una funzione modulatrice $G_\gamma(i)$, con un andamento definito in 3.25, la quale serve a cambiare il threshold α durante le iterazioni.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`bad_th`	reale positivo	3	imposta il threshold δ usato nelle confidenze LRC e MED
`conv_low_th`	$[0.0, 1.0]$	0.2	imposta il threshold α_L percentile per la conversione delle confidenze reali
`conv_high_th`	$[0.0, 1.0]$	0.8	imposta il threshold α_H percentile per la conversione delle confidenze reali
`conf_th`	$[0.0, 1.0]$	0.2	imposta il threshold γ per la funzione modulatrice

Tabella 4.4: iperparametri della confidenza UCM.

Durante l'implementazione è stato effettuato un *porting* Python dal sorgente originale[2], tuttavia è risultata troppo lenta.
UCM è stata riscritta in Cython ed è stata ottimizzata sfruttando il vincolo stringente (Eq. 4.17): ogni misura interna successiva alla prima viene calcolata solamente nei punti dove c'è già consenso da parte delle misure precedenti.

[2]Link al sorgente: `https://github.com/fabiotosi92/Unsupervised-Confidence-Measures`

Local Global Confidence Network

La misura di confidenza *Local Global Confidence (LGC)* [33] è realizzata tramite una rete neurale complessa composta da tre reti neurali indipendenti. L'idea che contraddistingue la rete è la fusione di confidenze *deep-local* (LFN) e confidenze *deep-global* (ConfNet).

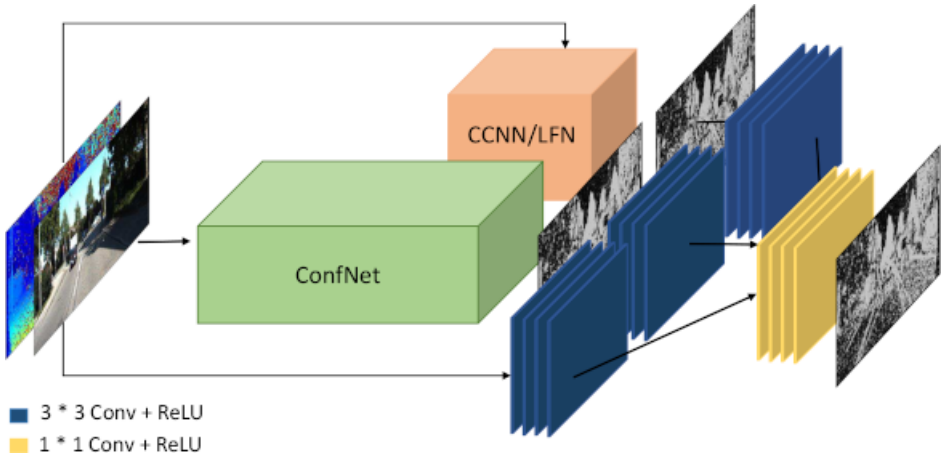


Figura 4.13: architettura della misura di confidenza LGC[3]: le reti ConfNet e LFN rappresentano rispettivamente la confidenza globale e la confidenza locale.

- La rete Late Fusion Network (LFN) si occupa della parte locale, cioè stimare la confidenza in un punto **m** solamente con le informazioni locali tramite una piccola finestra di supporto attorno al pixel.

  La rete utilizza un'architettura simile a MC-CNN con all'ingresso la mappa di disparità e l'immagine reference, tuttavia i pesi sono separati dato che trattano domini diversi.

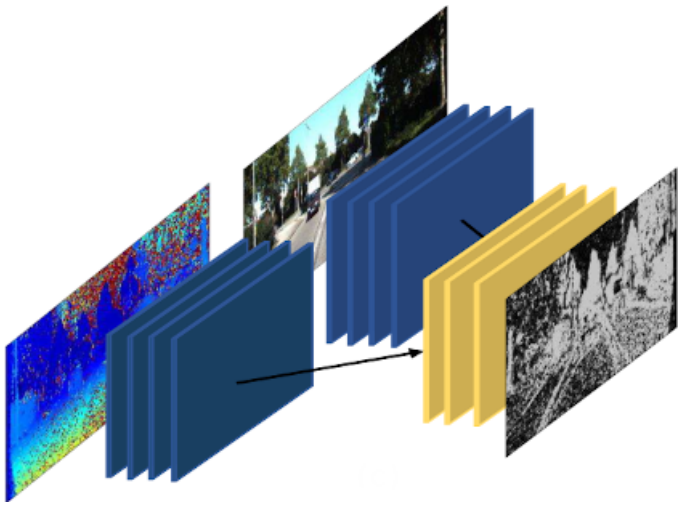


Figura 4.14: architettura della misura di confidenza LFN[3]: gli strati convoluzionali sono organizzati in modo simile a MC-CNN [18].

- La rete ConfNet si occupa della parte globale, cioè stimare la confidenza in un punto **m** usando informazioni provenienti da tutta l'immagine o da un sottoinsieme molto grande.

  La rete utilizza un'architettura piramidale, concettualmente simile a PyD-Net (Fig. 1.12), con all'ingresso la mappa di disparità e l'immagine reference: permette di raccogliere più informazioni nell'intorno del pixel dato che la finestra di supporto si allarga per effetto piramidale, tuttavia regolarizza la mappa delle confidenze riducendo i piccoli dettagli.

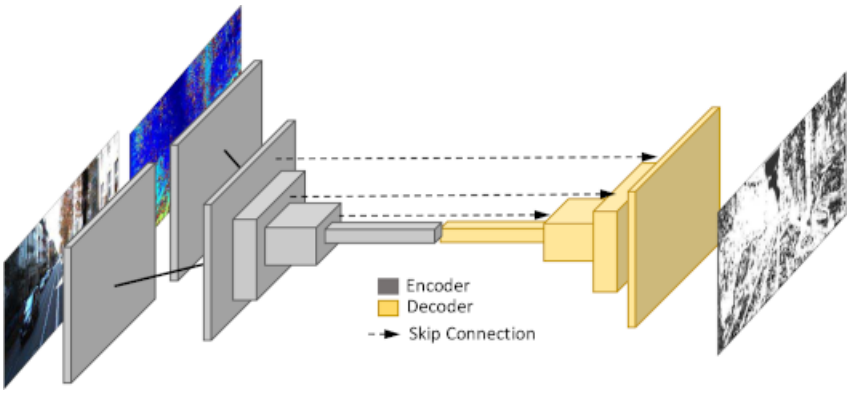


Figura 4.15: architettura della misura di confidenza ConfNet[3]: si può notare l'architettura piramidale *encoder-decoder* e gli *skip-connection* per mantenere dettaglio nell'*upscaling*.

- La rete LGC che si occupa di fondere le confidenze globali e locali, anche tramite le informazioni ricavate dalla mappa di disparità: l'architettura è composta da tre torri convoluzionali con pesi separati per i tre domini.

La rete LGC è stata già addestrata tramite dataset KITTI: gli effetti della proiezione non sono stati inclusi nell'addestramento.

Inoltre si tratta di una confidenza reale $c_{\mathcal{R}}(\mathbf{m}) \in [0.0, 1.0]$: non è obbligatorio convertire la misura reale in una misura binaria, tuttavia conviene per selezionare i migliori punti. Oltre che alla metodologia di conversione vista precedentemente (Eq. 4.16), è stata implementata una conversione più semplice a threshold senza percentile.

$$c^{+}(\mathbf{m}) = \begin{cases} 1 & \text{se } c_{\mathcal{R}}(\mathbf{m}) \geq \alpha_H \\ 0 & \text{altrimenti} \end{cases} \qquad c^{-}(\mathbf{m}) = \begin{cases} 1 & \text{se } c_{\mathcal{R}}(\mathbf{m}) \leq \alpha_L \\ 0 & \text{altrimenti} \end{cases} \tag{4.18}$$

[3]Fonte immagini architetture: [33].

Come in UCM, esiste la funzione modulatrice $G_\gamma(i)$ per cambiare i threshold α_L e α_H durante le iterazioni.

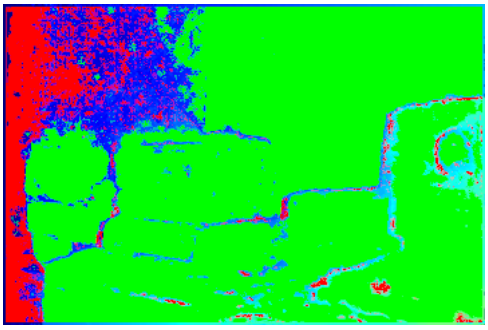


Figura 4.16: confidenza LGC, con $\alpha_L = 0.1$, $\alpha_H = 0.6$ e modalità a thresholding semplice.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`th_mode`	booleano	falso	falso per utilizzare il thresholding semplice (Eq. 4.18) e vero per utilizzare il thresholding percentile (Eq. 4.16)
`low_th`	$[0.0, 1.0]$	0.3	imposta il threshold α_L per la conversione a misura binaria
`high_th`	$[0.0, 1.0]$	0.6	imposta il threshold α_H per la conversione a misura binaria
`conf_th`	$[0.0, 1.0]$	0.2	imposta il threshold γ per la funzione modulatrice

Tabella 4.5: iperparametri della confidenza LGC.

LGC è stata implementata tramite un adattamento dal codice Python originale[4]: la rete non lavora più in modalità *standalone*, bensì è integrata all'interno del blocco di confidenza.

[4]Link al sorgente: `https://github.com/fabiotosi92/LGC-Tensorflow`

Self Adapting Confidence

Nell'articolo *Self Adapting Confidence* [34] non si propone una nuova misura di confidenza: si tratta di una strategia (chiamata *Out of The Box (OTB)*) per addestrare una misura di confidenza basata su rete neurale dopo il *deployment* nell'ambiente.

Come già discusso nella sottosezione 2.3.3, il grande vantaggio della strategia *self-learning* è l'adattamento al *domain-shifting*: la strategia può essere utile per adattare la confidenza alla proiezione virtuale, oltre che al nuovo ambiente.

Per funzionare la strategia ha bisogno di dati robusti da ricavare durante l'utilizzo della misura di confidenza: per ottenere la massima generalità sulle varie misure (i.e., modelli *black-box*, *gray-box* e *white-box*), occorre lavorare solamente sulla mappa di disparità reference e la coppia di immagini rettificate.

Per ottenere dati robusti e mantenere la massima generalità sono stati utilizzati tre criteri.

- Errore di riproiezione sull'immagine: si utilizza la mappa di disparità d^* per proiettare le intensità I_T dell'immagine target $\mathbf{I}_T$ nelle coordinate dell'immagine reference $\mathbf{I}_R$, ottenendo una nuova immagine $\hat{\mathbf{I}}_R$ con intensità di colore $\hat{I}_R$.

$$\begin{aligned} &\left\{\hat{\mathbf{m}}_R \in \hat{\mathbf{I}}_R,\, \mathbf{m}_R \in \mathbf{I}_R,\, \mathbf{m}_T \in \mathbf{I}_T,\, \mathbf{m}_R \leftrightarrow \mathbf{m}_T \;\middle|\right. \\ &\left.\hat{\mathbf{m}}_R = \begin{bmatrix} u_T + d^*(\mathbf{m}_R) \\ v_T \end{bmatrix},\; \hat{I}_R(\hat{\mathbf{m}}_R) = I_T(\mathbf{m}_T)\right\} \end{aligned} \tag{4.19}$$

  Successivamente si confrontano le intensità $\hat{I}_R$ e I_R per calcolare l'errore di riproiezione.

$$\Delta_\alpha(\mathbf{m}) = \alpha \cdot (1 - \mathrm{SSIM}_{(I_R, \hat{I}_R)}(\mathbf{m})) + (1 - \alpha) \left| I_R(\mathbf{m}) - \hat{I}_R(\mathbf{m}) \right| \tag{4.20}$$

  Dove SSIM è una misura di similarità basata su finestre di supporto attorno al punto $\mathbf{m}$.

- Consenso dei vicini sulla disparità: è una misura di confidenza che confronta le disparità del vicinato $\mathcal{N}_\mathbf{m}$ rispetto al punto centrale $\mathbf{m}$.

$$\mathrm{DA}(\mathbf{m}) = \frac{\mathcal{H}(d^*(\mathbf{m}))}{|\mathcal{N}_\mathbf{m}|} \tag{4.21}$$

Dove $\mathcal{H}(d)$ è la funzione d'istogramma al variare della disparità $d \in [d_{\min}, d_{\max}]$ sulla finestra di supporto $\mathcal{N}_{\mathbf{m}}$.

- Uniqueness Constraint (UC): a differenza di LRC richiede solamente la disparità reference.

I criteri verranno utilizzati per addestrare la misura di confidenza basata su reti neurali: è stata utilizzata una versione alleggerita di ConfNet, già addestrata su dataset KITTI.

Inoltre essendo ConfNet una confidenza reale $c_{\mathcal{R}}(\mathbf{m}) \in [0.0, 1.0]$, è consigliato trasformarla in confidenza binaria (Eq. 4.16) (Eq. 4.18).

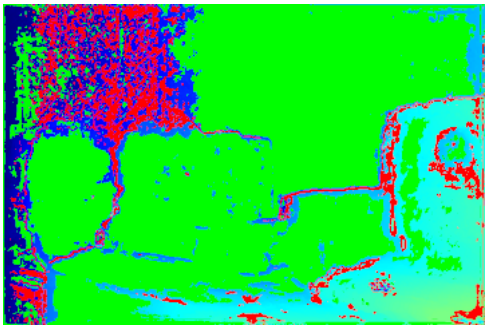


Figura 4.17: misura di confidenza ConfNet con strategia OTB, con $\alpha_L = 0.1$, $\alpha_H = 0.4$ e modalità a thresholding percentile.

Come in LGC, esiste la funzione modulatrice $G_\gamma(i)$ per cambiare i threshold α_L e α_H durante le iterazioni.

Iperparametri	**Tipo**	**Default**	**Descrizione**
`th_mode`	booleano	falso	falso per utilizzare il thresholding semplice (Eq. 4.18) e vero per utilizzare il thresholding percentile (Eq. 4.16)
`low_th`	$[0.0, 1.0]$	0.3	imposta il threshold α_L per la conversione a misura binaria
`high_th`	$[0.0, 1.0]$	0.6	imposta il threshold α_H per la conversione a misura binaria
`conf_th`	$[0.0, 1.0]$	0.2	imposta il threshold γ per la funzione modulatrice

Tabella 4.6: iperparametri della confidenza ConfNet con strategia OTB.

La strategia OTB è stata implementata assieme alla rete ConfNet tramite un adattamento dal codice Python originale[5]: la rete non lavora più in modalità *standalone*, bensì è integrata all'interno del blocco di confidenza.

4.1.5 Misure d'errore

Lo stadio delle misure è stato modificato per misurare anche l'errore di classificazione delle misure di confidenza binarie, basandosi sempre sulla ground-truth $d_g^*(\mathbf{m})$ fornita dal blocco dataset e allo stadio di confidenza per attingere alle confidenze positive $c^+(\mathbf{m})$ e negative $c^-(\mathbf{m})$.

Sono state implementate le misure classiche d'errore di classificazione (i.e., accuratezza, precisione e sensibilità), basate su *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)*, *False Negative (FN)*.

- L'accuratezza (*accuracy*) indica la percentuale di previsioni corrette (TP+TN) su tutte le previsioni eseguite.

$$\text{ACCURACY} = \frac{\text{TP} + \text{TN}}{\text{N}} \tag{4.22}$$

- La precisione (*precision*) indica la percentuale di previsioni positive corrette (TP) su tutte le previsioni positive (TP+FP), nel nostro caso se otteniamo grande precisione allora $c^+(\mathbf{m})$ prevede correttamente le confidenze positive.

$$\text{PRECISION} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.23}$$

- La sensibilità (*recall*) indica la percentuale di previsioni positive corrette su (TP) su tutte le classificazioni effettivamente positive (TP+FN), nel nostro caso se otteniamo grande sensibilità allora $c^+(\mathbf{m})$ prevede tutti i casi di confidenze positive.

$$\text{RECALL} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.24}$$

Definiamo anche TP, TN, FP, FN associati alle misure di confidenze $c^+(\mathbf{m})$ e $c^-(\mathbf{m})$.

$$\text{TP} = \sum_{\mathbf{m} \in \mathbf{I}} \Lambda^+(\mathbf{m}) \quad \Lambda^+(\mathbf{m}) = \begin{cases} 1 & \text{se } c^+(\mathbf{m}) = 1 \wedge \left| d^*(\mathbf{m}) - d_g^*(\mathbf{m}) \right| \leq \delta \\ 0 & \text{altrimenti} \end{cases} \tag{4.25}$$

[5]Link al sorgente: `https://github.com/mattpoggi/self-adapting-confidence`

$$\mathrm{TN} = \sum_{\mathbf{m}\in\mathbf{I}} \Lambda^{-}(\mathbf{m}) \quad \Lambda^{-}(\mathbf{m}) = \begin{cases} 1 & \text{se } c^{-}(\mathbf{m}) = 1 \wedge \left|d^{*}(\mathbf{m}) - d_{g}^{*}(\mathbf{m})\right| > \delta \\ 0 & \text{altrimenti} \end{cases} \tag{4.26}$$

$$\mathrm{FP} = \sum_{\mathbf{m}\in\mathbf{I}} \lambda^{+}(\mathbf{m}) \quad \lambda^{+}(\mathbf{m}) = \begin{cases} 1 & \text{se } c^{+}(\mathbf{m}) = 1 \wedge \left|d^{*}(\mathbf{m}) - d_{g}^{*}(\mathbf{m})\right| > \delta \\ 0 & \text{altrimenti} \end{cases} \tag{4.27}$$

$$\mathrm{FN} = \sum_{\mathbf{m}\in\mathbf{I}} \lambda^{-}(\mathbf{m}) \quad \lambda^{-}(\mathbf{m}) = \begin{cases} 1 & \text{se } c^{-}(\mathbf{m}) = 1 \wedge \left|d^{*}(\mathbf{m}) - d_{g}^{*}(\mathbf{m})\right| \leq \delta \\ 0 & \text{altrimenti} \end{cases} \tag{4.28}$$

$$\mathrm{N} = \mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN} \tag{4.29}$$

Iperparametri	**Tipo**	**Default**	**Descrizione**
`th_conf`	reale	3	imposta δ per calcolare TP,TN,FP,FN
`zero_check_conf`	booleano	falso	vero se si vuole confrontare anche le disparità invalide ($d^{*} = 0$)

Tabella 4.7: iperparametri aggiornati dello stadio delle misure d'errore.

Tutte le misure d'errore aggiuntive sono state ottimizzate tramite Cython per ottenere una valutazione dei risultati più veloce.

4.2 Valutazione della strategia modificata

Per verificare l'efficacia delle misure di confidenza per attingere a nuovi punti affidabili, sono stati effettuati una serie di test quantitativi, analizzati successivamente con grafici e schemi.

Anche in questo caso il task che ha richiesto più tempo è stato la ricerca degli iperparametri ottimali: le misure di confidenza richiedono tempo aggiuntivo per essere calcolate. L'incremento di tempo è dovuto anche all'algoritmo di matching applicato ad ogni iterazione, oltre che alla misura di confidenza.

4.2.1 Valutazione qualitativa

La valutazione qualitativa si focalizza nell'analizzare i risultati della ricerca dei migliori iperparametri di proiezione a seconda della misura di confidenza.

- confidenze negative nel DSI: l'applicazione della funzione modulatrice $G^{-}_{\mathbf{m}}(d)$ (Eq. 4.4) è utile solamente nelle confidenze ad alta sensibilità (i.e., LGC e oracolo), dato che un aumento del costo può penalizzare una corrispondenza in realtà giusta;
- threshold semplice: può essere applicato solamente alle misure più accurate (i.e., LGC e oracolo), perché la soglia percentile effettua anche una selezione dei migliori punti;
- finestra del consenso: il meccanismo del consenso (Fig. 4.3) è utile solamente nelle misure più rumorose (i.e., UCM, OTB+ConfNet), dato che potrebbe nascondere un cambiamento effettivo da confidenza negativa a confidenza positiva (i.e., la confidenza ha portato alla scoperta di un punto affidabile);
- rumore Gaussiano adattivo: si è rivelato efficace solamente nella confidenza oracolo poiché l'effetto di regolarizzazione può ingannare la misura, specialmente per le misure a rete neurale dato che non sono abituate all'uso del rumore sulle immagini;
- funzioni modulatrici: le funzioni modulatrici non hanno un impatto significativo per il rumore Gaussiano adattivo e per la dimensione dei punti.

4.2.2 Valutazione quantitativa

La valutazione quantitativa consiste in un ablation study dove si mettono a confronto SGM tradizionale rispetto a SGM+VPP: vengono valutati i punti affidabili forniti dalle varie misure di confidenza.

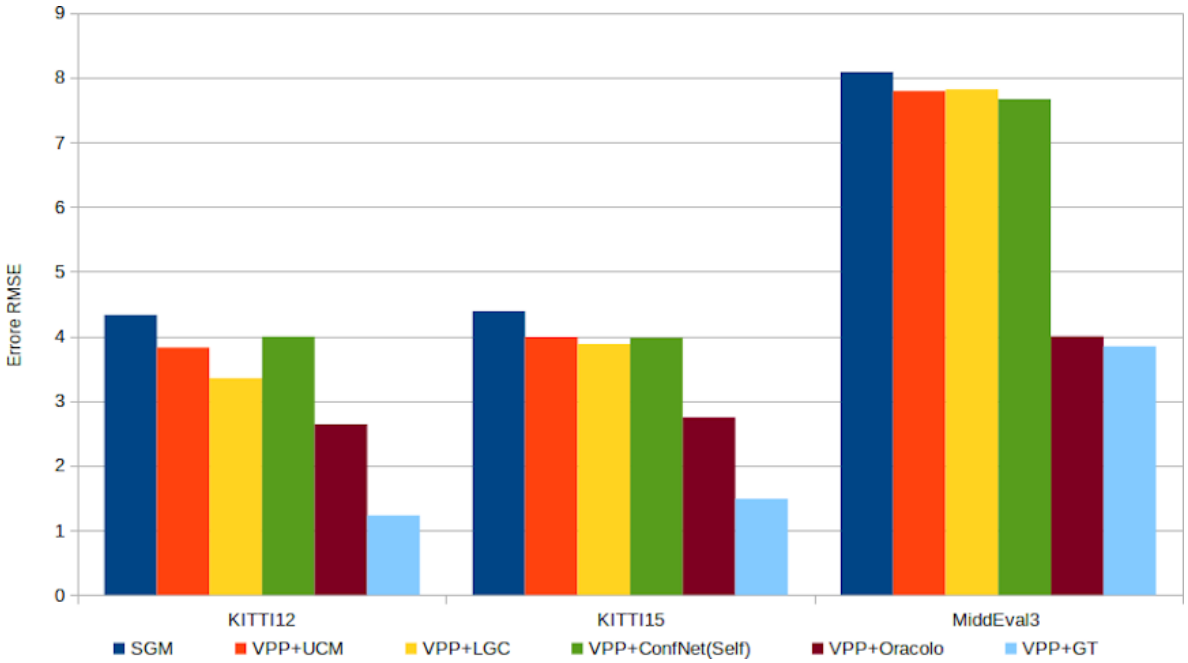


Figura 4.18: risultati qualitativi della strategia SGM+VPP sui dataset KITTI12 [25], KITTI15 [26], MiddEval3 [5] usando solo le misure di confidenza come fonte di punti affidabili: nel grafico è incluso un confronto con l'oracolo ( ) e con punti affidabili da ground-truth ( ).

Variazione percentuale rispetto a SGM - tecnica SGM+VPP				
Dataset	**UCM**	**LGC**	**OTB+ConfNet**	**Oracolo**
KITTI12	-12%	-23%	-8%	-39%
KITTI15	-9%	-12%	-9%	-37%
MiddEval3	-4%	-3%	-5%	-51%

Tabella 4.8: variazione percentuale dell'errore RMSE rispetto alla tecnica tradizionale SGM: si utilizza la tecnica di proiezione VPP con misure di confidenza differenti.

I risultati (Fig. 4.18, Tab. 4.8) mostrano che i punti affidabili forniti dalle misure di confidenza non sono cruciali per ottenere un miglioramento significativo dell'errore.
Si distinguono le misure di confidenze a rete neurale LGC e OTB+ConfNet: la prima ottiene un ottimo risultato sul dataset di addestramento anche se perde efficacia nei dataset mai visti (domain-shifting), mentre la seconda ottiene performance più scadenti ma efficaci in tutti i dataset.

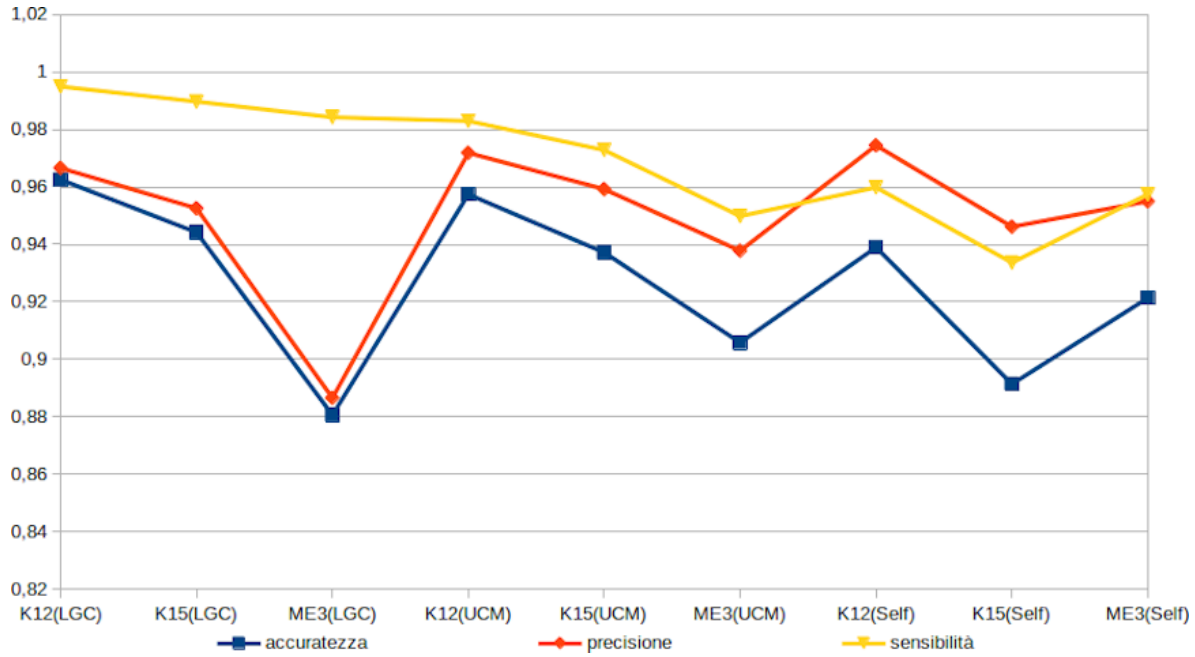


Figura 4.19: accuratezza ( ), precisione ( ) e sensibilità ( ) delle misure di confidenza LGC,UCM e OTB+ConfNet nei dataset KITTI12 [25], KITTI15 [26], MiddEval3 [5].

Dalla figura 4.19 si può notare che LGC ha un'ottima sensibilità sul dataset di addestramento, ma perde precisione negli altri dataset, mentre OTB+ConfNet mantiene una precisione costante, ma una sensibilità più bassa.
Inoltre si nota una correlazione tra accuratezza e riduzione dell'errore.

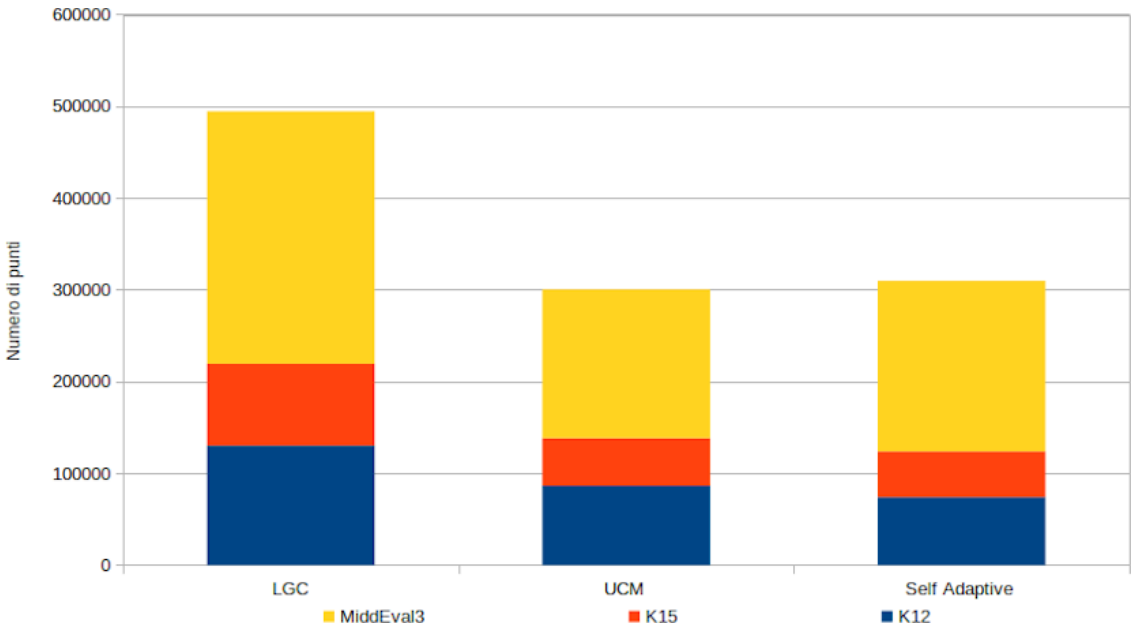
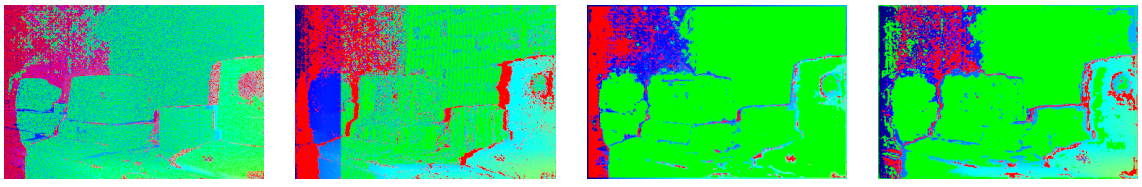


Figura 4.20: punti classificati dalle misure di confidenza LGC,UCM e OTB+ConfNet nei dataset KITTI12 [25](■), KITTI15 [26](■), MiddEval3 [5](■).

Infine dalla figura 4.20 si può notare l'effetto del thresholding semplice (applicato a LGC) rispetto al thresholding percentile (applicato a OTB+ConfNet e UCM): il thresholding percentile taglia fuori una buona parte dei punti classificati, perché considerati incerti.

4.2.3 Fallimenti

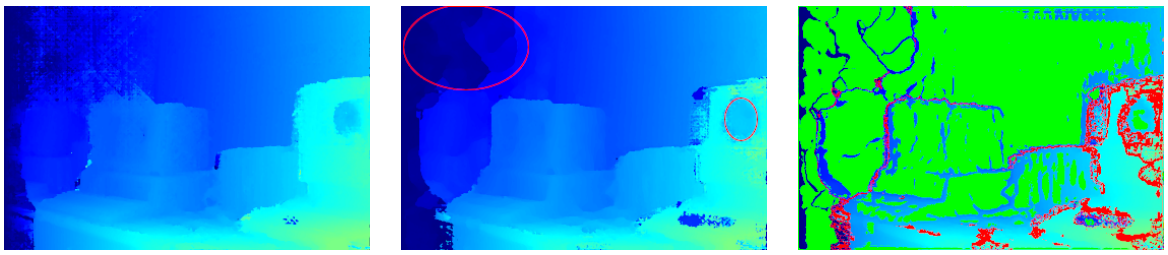
Osservando i risultati delle confidenze reali possiamo constatare che i fallimenti delle misure sono collegati alle principali problematiche della corrispondenza.



Oracolo UCM LGC OTB+ConfNet

Figura 4.21: confronto della confidenza perfetta con le confidenze reali: si notano errori nelle superifici non lambertiane e nelle regioni uniformi.

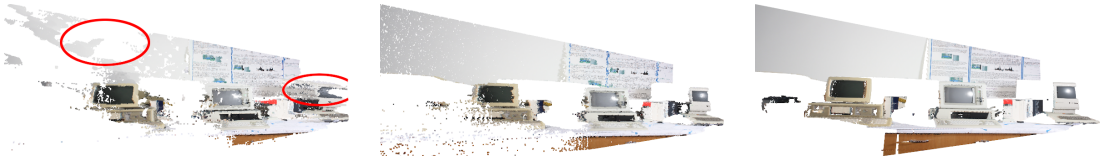
I fallimenti dovuti alle problematiche della corrispondenza portano ad una incertezza nelle superfici non lambertiane e nelle zone uniformi (Fig. 4.21), i quali causano errori nella proiezione.



prima iterazione | ultima iterazione | confidenza distorta

Figura 4.22: esempio di applicazione della misura di confidenza OTB+ConfNet insieme alla tecnica VPP: l'accumulo di errori porta ad una distorsione che non viene rilevata dalla misura.

Gli errori nella proiezione si accumulano, portando ad un risultato distorto rispetto alla scena reale (Fig. 4.22, Fig. 4.23).



OTB+ConfNet | VPP+Guided | ground-truth

Figura 4.23: comparazione delle point cloud: l'accumulo di errori OTB+ConfNet porta ad una distorsione del muro, mentre nel metodo proposto nel Capitolo 3 non c'è accumulo d'errore dato che i punti affidabili rimangono immutati.

Per ridurre il problema senza ricorrere a sensori attivi, potrebbe essere utile l'introduzione della segmentazione semantica, per esempio tramite un sensore monoculare.

Conclusioni

Dopo aver introdotto le tecnologie riguardo la percezione della profondità tra cui i sensori passivi stereo e la loro teoria (i.e., modello stereo laterale), viene descritto lo stato dell'arte attuale ed esposta la nuova strategia di fusione basata sulla proiezione virtuale. La soluzione a proiezione ha richiesto l'implementazione di componenti aggiuntivi nella pipeline stereo tradizionale (i.e., coordinatore, proiettore, integratore del volume di costo e sorgente di punti affidabili).

Nonostante lo sforzo di ricerca e implementazione, le valutazioni eseguite nel Capitolo 3 hanno confermato l'efficacia della soluzione a proiezione surclassando la precedente tecnica allo stato dell'arte ([7]).
Oltre ai successi sono stati esposti anche i limiti della metodologia a proiezione virtuale (i.e., perdita di dettagli fini): questi limiti possono essere affrontati in uno sviluppo futuro, grazie ai suggerimenti proposti.

Infine sono state utilizzate le misure di confidenza come fonte di punti affidabili per analizzare la fattibilità di un'eventuale indipendenza da sensori attivi per la proiezione.
Le valutazioni eseguite nel Capitolo 4 hanno evidenziato che tale soluzione sia attualmente limitante, mostrando i limiti delle misure dovuti soprattutto agli errori accumulati durante le iterazioni.
Ciononostante il campo di ricerca che riguarda le misure di confidenza è ancora aperto: le misure di confidenza basate su rete neurale utilizzate non erano preparate a gestire gli effetti della proiezione e possono essere addestrate usando una strategia che preveda l'utilizzo della proiezione virtuale.

Acronimi

API Application Programming Interface. 99

APKR Average PeaK Ratio. 81

BLOB Binary Large OBject. 4

BM Block Matching. 62

BMP Bad Matching Pixel. 64

BMPRE Bad Matching Pixel Relative Error. 64

CMYK Cyan Magenta Yellow Key. 2

CPU Central Processing Unit. 65

CRF Camera Reference Frame. 3, 12, 15, 17, 18, 99

cwToF continuous wave Time of Flight. iv, 5, 7, 8

DLB Distance to Left Border. 80

DSI Disparty Space Image. 27, 75, 90

FN False Negative. 88, 89

FP False Positive. 88, 89

FW Fixed Window. 27, 37

GC-Net Geometry and Context Network. 30

gpGPU general purpose Graphics Processing Unit. 65

iDSI integrated Disparty Space Image. 35

IR Infrared Radiation. 8

LFN Late Fusion Network. 83

LGC Local Global Confidence. 83–85, 87, 90–93

LiDAR Light Detection And Ranging. iv, v, 5–7, 40

LoG Laplancian of Gaussian. 41

LRC Left Right Consistency. 79, 82, 87

MC-CNN Matching Cost - Convolutional Neural Network. 30, 83

MED MEdian Deviation of disparity. 80, 82

MRE Mean Relative Error. 63

MSE Mean Square Error. 63

OTB Out of The Box. 86–88, 90–94

PPM Perspective Projection Matrix. 15, 17, 18, 20, 21, 26, 34

RANSAC RANdom SAmple Consensus. 12

RGB Red Green Blue. 2, 3, 34, 99

RMSE Root Mean Square Error. 19, 63, 64, 67, 68, 91

rSGM rapid Semi Global Matching. 37, 41, 43, 62, 63

RTT Round Trip Time. 5, 7

SAD Sum of Absolute Differences. 27

SGM Semi Global Matching. 27–29, 35, 37, 41, 62, 63, 66–70, 91

SIMD Single Istruction Multiple Data. 37, 42, 43, 63

SLAM Simultaneous Localization And Mapping. 4, 6

SNR Signal to Noise Ratio. 6

SO Scanline Optimization. 28, 29

SRF Stereo Reference Frame. 3, 12, 15, 21, 26, 34

SSD Sum of Square Differences. 27

SSIM Structural Similarity Index Measure. 86

TN True Negative. 88, 89

ToF Time of Flight. iv, v, 5–7

TP True Positive. 88, 89

UC Uniqueness Constraint. 79, 87

UCM Unsupervised Confidence Measures. 79, 82, 85, 90–93

VPP Virtual Pattern Projection. 33, 61, 66–69, 71, 91, 94

WMN Winner MargiN. 80, 81

WRF World Reference Frame. 12, 15, 17, 18

Glossario

ablation study studio in cui si confronta un'entità con se stessa, ma senza determinate caratteristiche/funzioni. 66–68, 91

biomimetica disciplina che studia e imita la natura come fonte di ispirazione e miglioramento delle tecnologie. iv

canale contiene i dati dell'intensità luminosa, che può essere rappresentativa di una scala di grigi o di un singolo colore. 2

colormap funzione che trasforma le intensità di grigi a certi colori prefissati di un modello di colore (e.g., RGB), per una migliore rappresentazione visiva. 3, 22

Cython permette di velocizzare il codice Python, compilandolo in codice nativo. 38, 46, 54, 57, 58, 60, 64, 82, 89

mappa di disparità immagini a scala di grigi dove ogni pixel codifica il valore di disparità in un'intensità luminosa. v, 4, 9, 22, 23, 28, 35, 36, 39, 40, 55, 59, 61, 63, 72, 83, 84, 86, 99

mappa di profondità immagine a scala di grigi dove ogni pixel codifica la distanza rispetto ad un riferimento (i.e., CRF). v, 3, 4, 22

Matplotlib libreria per il disegno di grafici sviluppata per Python. 38

modello *black-box* l'algoritmo di matching si comporta come una scatola nera: nessuna informazione oltre che alla mappa di disparità viene fornita. 35, 86

modello *gray-box* l'algoritmo di matching espone delle informazioni aggiuntive tramite API. 86

modello *white-box* l'algoritmo di matching espone le strutture dati intermedie, tra cui il volume di costo se previsto. 35, 86

NumPy libreria matematica sviluppata per Python. 38, 60, 74, 76

OpenCV libreria *open-source* per lo sviluppo di software legato alla Computer Vision. 18, 19, 38

pesi nel campo delle reti neurali artificiali, i pesi rappresentano la conoscenza appresa dalla rete. 83, 84, 100

proiezione virtuale fotoritocco di una scena osservata tramite punti 3D: i punti 2D sono disegnati su entrambe le viste e mantengono i vincoli geometrici. v, 33, 34, 36, 44, 55, 86, 95

Python linguaggio di alto livello flessibile e adatto alla prototipizzazione, largamente usato in ambiti scientifici. 38, 61, 74, 76, 82, 85, 88, 99, 100

Ray libreria avanzata per il calcolo parallelo sviluppata per Python. 38

rete siamese rete neurale artificiale che usa gli stessi pesi per elaborare due input differenti. 30

segmentazione semantica permette di partizionare l'immagine in categorie con un certo valore semantico (e.g., "cassa", "tavolo", "primo piano", "uniforme"). 4, 94

stringa di bit sequenza di bit con un ordine prestabilito. 41–43, 51

superficie lambertiana superficie ideale che riflette la luce incidente da una direzione in modo uguale in tutte le direzioni (cambiando punto di vista si ha la stessa luminosità). 24, 32, 35, 36, 48, 49, 55, 93

Tensorflow libreria multipiattaforma utilizzata per la creazione, l'addestramento e l'esecuzione di reti neurali profonde. 38

Bibliografia

[1] Filippo Aleotti, Giulio Zaccaroni, Luca Bartolomei, Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Real-time single image depth perception in the wild with handheld devices. *Sensors*, 21(1), 2021.

[2] Stefano Mattoccia. Stereo vision: Algorithms and applications. DISI - University of Bologna, 2013.

[3] Richard Szeliski. *Computer Vision - Algorithms and Applications, Second Edition.* Texts in Computer Science. Springer, 2022.

[4] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2019.

[5] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014.

[6] Yongjun Zhang, Siyuan Zou, Xinyi Liu, Xu Huang, Yi Wan, and Yongxiang Yao. LiDAR-guided stereo matching with a spatial consistency constraint. *ISPRS Journal of Photogrammetry and Remote Sensing*, 183:164–177, jan 2022.

[7] Matteo Poggi, Davide Pallotti, Fabio Tosi, and Stefano Mattoccia. Guided stereo matching, 2019.

[8] Li Zhang, B. Curless, and S.M. Seitz. Spacetime stereo: shape recovery for dynamic scenes. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–367, 2003.

[9] R W G Hunt. *The reproduction of colour.* The Wiley-IS&T Series in Imaging Science and Technology. Wiley-Blackwell, Hoboken, NJ, 6 edition, September 2004.

[10] Jingyun Liu, Qiao Sun, Zhe Fan, and Yudong Jia. Tof lidar development in autonomous vehicle. In *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, pages 185–190, 2018.

[11] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Patrice Horaud. *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media, 2012.

[12] Joaquim Salvi, Jordi Pagès, and Joan Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, 2004. Agent Based Computer Vision.

[13] Federico Tombari. Algoritmi per la corrispondenza stereo. Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), 2008.

[14] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5848–5854. IEEE, 2018.

[15] Luigi Di Stefano. Image formation and acquisition. DISI - CVLab - University of Bologna, 2021.

[16] Luigi Di Stefano and Samuele Salti. Image formation and acquisition – camera calibration. DISI - CVLab - University of Bologna, 2021.

[17] Luigi Di Stefano. Principi ed applicazioni della visione stereo. DEIS/ARCES - University of Bologna, 2008.

[18] Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, 2016.

[19] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE international conference on computer vision*, pages 66–75, 2017.

[20] Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *2021 International Conference on 3D Vision (3DV)*, pages 218–227. IEEE, 2021.

[21] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks

for disparity, optical flow, and scene flow estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[22] Matteo Poggi, Alessio Tonioni, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Continual adaptation for deep stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[23] Pierluigi Zama Ramirez, Fabio Tosi, Matteo Poggi, Samuele Salti, Stefano Mattoccia, and Luigi Di Stefano. Open challenges in deep stereo: the booster dataset, 2022.

[24] Robert Spangenberg, Tobias Langner, Sven Adfeldt, and Raúl Rojas. Large scale semi-global matching on the cpu. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 195–201. IEEE, 2014.

[25] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[26] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[27] Luigi Di Stefano. Spatial filtering. DISI - CVLab - University of Bologna, 2021.

[28] G. E. P. Box and Mervin E. Muller. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610 – 611, 1958.

[29] Christian Banz, Peter Pirsch, and Holger Blume. Evaluation of penalty functions for semi-global matching cost aggregation. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences; 39-B3*, 39(B3):1–6, 2012.

[30] Masao Shimizu and Masatoshi Okutomi. Sub-pixel estimation error cancellation on area-based matching. *International Journal of Computer Vision*, 63(3):207–224, 2005.

[31] Saad Merrouche, Milenko Andrić, Boban Bondžulić, and Dimitrije Bujaković. Objective image quality measures for disparity maps evaluation. *Electronics*, 9(10):1625, 2020.

[32] Fabio Tosi, Matteo Poggi, Stefano Mattoccia, Alessio Tonioni, and Luigi di Stefano. Learning confidence measures in the wild. In *BMVC*, 2017.

[33] Fabio Tosi, Matteo Poggi, Antonio Benincasa, and Stefano Mattoccia. Beyond local reasoning for stereo confidence estimation with deep learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 319–334, 2018.

[34] Matteo Poggi, Filippo Aleotti, Fabio Tosi, Giulio Zaccaroni, and Stefano Mattoccia. Self-adapting confidence estimation for stereo. In *European Conference on Computer Vision*, pages 715–733. Springer, 2020.

Ringraziamenti

Con questa tesi si conclude il mio percorso universitario da studente, iniziato cinque anni fa: è stato un percorso lungo e faticoso, ma con tanta passione e fede sono riuscito a portare a termine questo traguardo.
Quando scelsi di intraprendere la laurea in ingegneria informatica ero ignaro di quante sfide avevo di fronte: ogni ostacolo che mi si è posto davanti mi ha permesso di maturare sia professionalmente sia personalmente e di affrontare lo scoglio successivo. Volevo quindi ringraziare per primo questo ciclo di studi che mi ha forgiato in quello che sono.

Questi ultimi due anni sono stati i più difficili: il covid ha portato via le risate, gli amici e tutto il calore che animava le aule del dipartimento. Ho reagito a questa pandemia un po' come il ciclista all'ultimo chilometro, stringendo i denti e pensando che tra poco c'è l'arrivo: solo ora che sono in volata ad un metro dal traguardo ho capito l'aiuto che può dare un compagno di banco a vivere la vita e non solo quella universitaria. Ringrazio perciò il "gruppo serio", il "gruppo pendolari" e tutti gli amici di vecchia data per tutta la compagnia offerta prima della pandemia, ma anche tutte le conoscenze di dopo.

Ringrazio il Prof. Mattoccia per avermi mostrato un campo di mio interesse ed insieme ai Dott. Aleotti, Poggi e Tosi per avermi aiutato con questa tesi, con la tesi precedente e con la domanda di dottorato.
Infine volevo ringraziare la mia famiglia, anche chi non c'è più, per non avermi fatto mancare nulla e per la grande fiducia che mi avete dato.

Il proverbio dice che non tutto il male viene per nuocere: in questo periodo fuori dal comune ho adottato Mina, una gattina a cui io e tutta la famiglia vogliamo molto bene, ma sopratutto mi ha permesso di conoscere una persona che tutt'oggi continua a stare al mio fianco e a rendermi sempre più forte.