ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DISI

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

TESI DI LAUREA
in
Calcolatori Elettronici T

METODOLOGIE DI APPRENDIMENTO AUTOMATICO APPLICATE ALLA GENERAZIONE DI DATI 3D

CANDIDATO:
Matilde Ugolini

RELATORE:
Chiar.mo Prof. Stefano Mattoccia

CORRELATORE:
Dott. Matteo Poggi

Anno Accademico 2014/15

Sessione III

CAPITOLI

CAPITOLO 1

1 INTRODUZIONE

Il *framework* oggetto della tesi, sviluppato in una precedente tesi di laurea [7], è un ambiente ideato con lo scopo di applicare tecniche di Machine Learning (in particolare si sfrutterà la capacità di previsione delle Random Forest) alle funzionalità dell'algoritmo di stereo matching SGM (Semi Global Matching), al fine di incrementare l'accuratezza di tale algoritmo in versione standard.

Il framework si basa come già anticipato su due fondamentali concetti: la visione stereo e il machine learning.

La visione stereo è una tecnica ampiamente utilizzata in Computer Vision che, data una scena, mira a stimare la profondità basandosi su due o più immagini acquisite da altrettanti punti di vista differenti (i.e. mediante una stereo camera). Tale tecnica permette di ottenere una rappresentazione tridimensionale dello spazio pur avendo a disposizione soltanto delle semplici immagini in due dimensioni.

Per Machine Learning si intende invece quella branca dell'intelligenza artificiale che si occupa dello sviluppo di algoritmi e tecniche che consentono ai calcolatori di apprendere informazioni di qualsivoglia natura e di utilizzare la conoscenza appresa per la sintesi di nuova conoscenza, sia essa sotto forma di una classificazione o della previsione di un risultato.

L'unione di questi due mondi ha apportato notevoli migliorie a SGM standard, come mostrato in [7], ma nonostante questo si è sempre alla ricerca di metodi che possano ulteriormente apportare benefici.

Scopo della presente tesi è proprio quello di continuare tale ricerca, ampliando e ove necessario, modificando le funzionalità del framework precedentemente sviluppato e denominato in seguito SGM_Random_Forest.

CAPITOLO 2

2 STEREO VISION

La tecnica della Stereo Vision si basa sull'utilizzo di due o più camere che inquadrano una stessa scena da differenti punti di vista.

Nel caso in questione le immagini saranno acquisite tramite stereo camera in *forma standard*, quindi due camere idealmente allineate in modo perfetto e poste alla stessa altezza, condizioni che, oltretutto, più si avvicinano alla natura della vista umana.

Tramite l'elaborazione di queste due immagini, *reference* e *target*, è possibile ricostruire la profondità della scena, per via della particolare geometria del sistema stereoscopico, la *geometria epipolare*.

Grazie ad essa possiamo risolvere il problema della corrispondenza tra punti omologhi delle due immagini. In particolare, utilizzando la forma standard della stereo camera possiamo semplificare la ricerca di tale corrispondenza portando il problema da un piano bidimensionale ad uno ad una sola dimensione in quanto dalla teoria sappiamo che pixel omologhi giacciono sulla stessa *scanline*.

Le due camere sono poste ad una distanza T tra di loro, e osservano entrambe lo stesso punto ***P*** rispettivamente dai centri dei loro piani di proiezione $\boldsymbol{O_l}$ e $\boldsymbol{O_r}$. Per ciascuna di esse, ***P*** sarà proiettato sul rispettivo piano di proiezione nei punti $\boldsymbol{p_l}$ e $\boldsymbol{p_r}$, aventi medesime coordinate verticali e coordinate orizzontali $\boldsymbol{p_l}$ e $\boldsymbol{p_r}$.

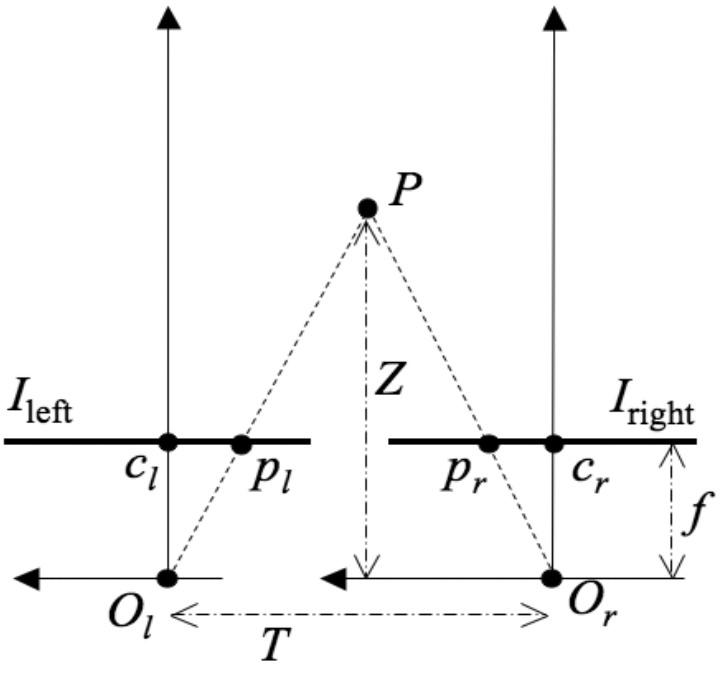


Figure 2.1: Geometria Stereoscopica [16]

E' possibile quindi valutare lo scostamento orizzontale, denominato disparità $\boldsymbol{d}$ tra pixel omologhi come

$$\boldsymbol{d} = \boldsymbol{p_l} - \boldsymbol{p_r}$$

Ciò ci permette di determinare, tramite una procedura di triangolazione e la conoscenza dei parametri del sistema stereoscopico (quali distanza T tra i due centri di proiezione e lunghezza focale delle due camere f), la posizione del punto nello spazio tridimensionale, cioè la profondità: più i due punti sono lontani, minore è la disparità e viceversa; minore è la disparità maggiore sarà la distanza di P dalla stereocamera.

$$\frac{T - (\boldsymbol{p_l} - \boldsymbol{p_r})}{Z - f} = \frac{T}{Z} = \frac{T \cdot f}{(\boldsymbol{p_l} - \boldsymbol{p_r})} = \frac{T \cdot f}{\boldsymbol{d}}$$

Individuando tutte le coppie di pixel omologhi delle immagini, la cosiddetta tecnica di matching stereo, e iterando il calcolo della disparità per tutte le coppie di pixel ottenute, si ottiene la mappa di disparità della scena in esame, ossia un'immagine in scala di grigi dove l'intensità di ciascun pixel è proporzionale alla disparità in quel punto (i pixel chiari sono i punti più vicini alle camere).

2.1 ALGORITMI DI MATCHING STEREO

Il compito dell'individuazione dei pixel omologhi nelle immagini Reference e Target e del calcolo della rispettiva disparità è affidato agli algoritmi di matching stereo.

L'idea generale alla base di tali algoritmi è quella di confrontare ogni pixel dell'immagine Reference con quelli dell'immagine Target e individuare così il corrispettivo pixel, in modo da trovare il punto reale della scena e calcolarne la profondità.

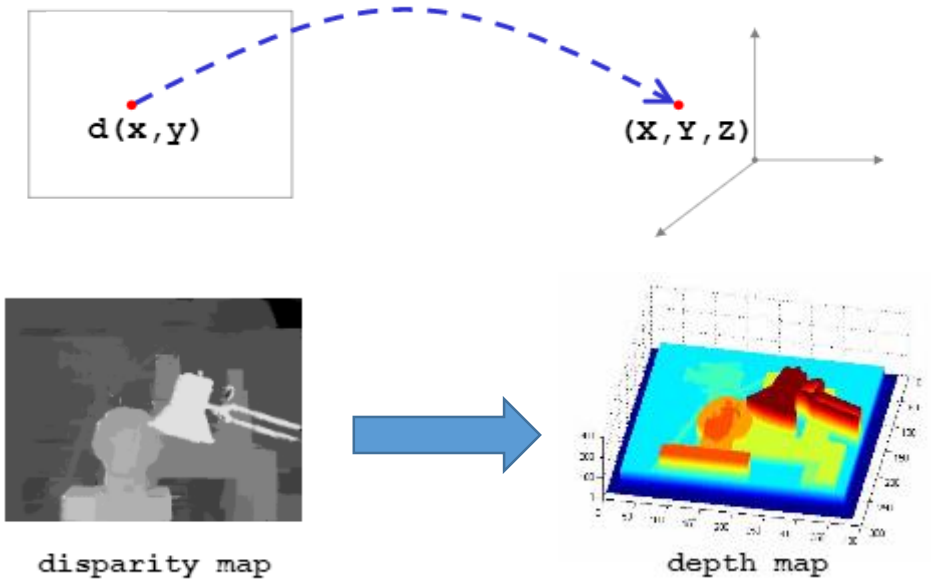


Figure 2.2: Calcolo della profondità di una scena [5]

L'approccio più semplice e inutilizzato per via della poca affidabilità dei risultati, è quello che prevede il confronto fra l'intensità dei pixel dell'immagine Reference di coordinate (x_i, y_i) con l'intensità dei pixel dell'immagine Target a coordinate (x_i + d, y_i), quindi posti alla stessa altezza, con d che rappresenta la disparità cercata, compreso fra 0 e dMax.

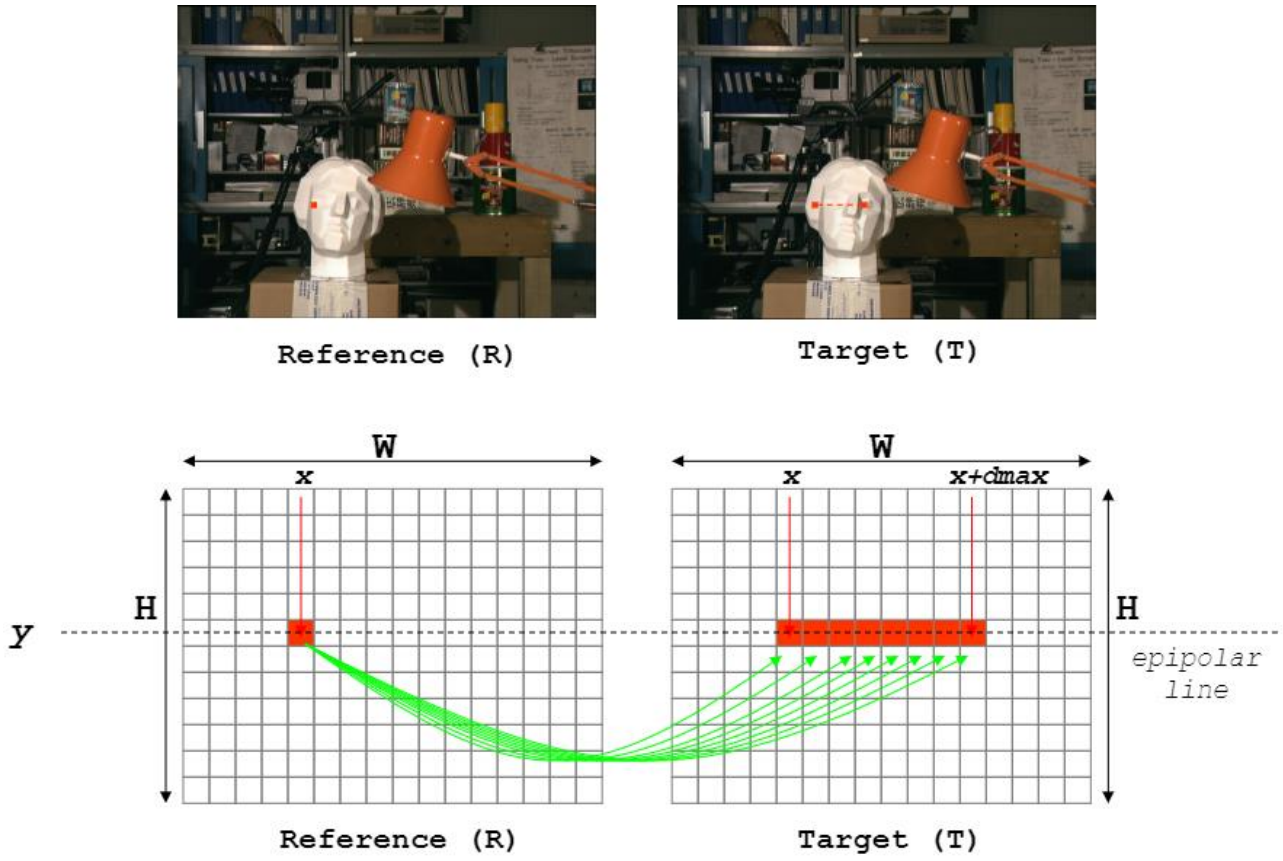


Figure 2.3: Algoritmo di matching basico [5]

A questo punto a ogni confronto viene assegnato un valore denominato costo, che indica quanto l'intensità del pixel Reference si discosta dal pixel Target con cui è stato confrontato (operazione di Matching Cost). La disparità ricercata d, sarà quella associata al costo minore dell'intervallo [0, dMax] relativo al pixel Reference in esame (*Winner-Takes-All Strategy*).

Gli algoritmi di matching stereo di suddividono in due principali categorie, non mutuamente esclusive, in base alla strategia implementata:

- Algoritmi Locali: adottano la strategia WTA per il calcolo della disparità, ma minimizzano le ambiguità calcolando i costi di matching su di una finestra di supporto che circonda il pixel di interesse. Si tratta di algoritmi veloci, ma le cui prestazioni sono messe in discussione da numerose problematiche, come aree con poca *texture* o con *pattern ripetitivi*, *occluding boundaries*.
- Algoritmi Globali: cercano la disparità che minimizzi una data funzione di energia sull'intera immagine. In particolare i globali ottengono risultati molto accurati, ma richiedono una potenza computazionale e una disponibilità di risorse non sempre abbordabili, specialmente se si vogliono implementare in applicazioni real-time o in sistemi embedded.

Un approccio alternativo è rappresentato dall'impostare il problema di minimizzazione della funzione energia in modo analogo all'approccio globale, ma riducendo la dimensionalità del problema. Questo permette di ottenere tecniche particolarmente veloci, vicine alle specifiche real-time. L'utilizzo di funzioni energia di tipo globale, invece, permette di avere più accuratezza, con risultati migliori delle tecniche locali tradizionali. Per questo per alcune di esse è stato coniato il termine semi-globale [1].

2.2 L'ALGORITMO SGM

L'algoritmo SGM [2], letteralmente Semi-Global Matching, è l'algoritmo che si è utilizzato nella realizzazione del framework SGM_Random_Forest, e implementa l'approccio alternativo ad algoritmi puramente locali o globali descritto nel capitolo precedente.

Come la maggior parte degli algoritmi stereo, e in particolare quelli globali, SGM

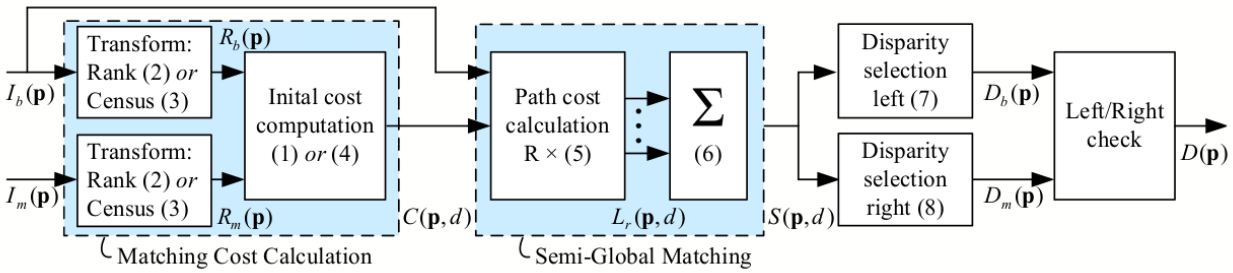


Figure 2.4: Fasi di elaborazione per la stima della disparità [3]

esegue i seguenti passaggi: [3]

- **Calcolo dei costi di matching:** Nell'implementazione di SGM scelta è utilizzata la differenza di Hamming calcolata in ogni punto delle trasformate census della coppia di immagini.
- **Aggregazione dei costi:** SGM effettua un'aggregazione dei costi provenienti da più punti dell'immagine e introduce una funzione di energia per evitare di produrre risultati ambigui, come può accadere invece se si basa il calcolo della disparità solo sui costi pixel per pixel.
- **Calcolo della disparità:** Al fine di calcolare il valore della disparità in ogni punto, sarà minimizzata la funzione di energia introdotta al punto precedente.

2.2.1 Calcolo dei Costi di Matching

Come anticipato, nell'implementazione di SGM scelta, si utilizza la differenza di Hamming calcolata sulla trasformazione *census* (census-Transformation) della coppia stereo.

La trasformata di census è una forma di trasformata locale non parametrica e consiste nel descrivere la porzione di un'immagine intorno ad un punto attraverso una stringa di bit avente tanti bit quanti sono i pixel della finestra di trasformazione meno il pixel preso in esame posto al centro. La stringa di bit è del tipo:

$$C[I(p)] = \bigodot_{p' \in S(p,\beta)} \xi(I, p, p')$$

dove $S(p, \beta)$ rappresenta la finestra di trasformazione, di raggio β centrata in $\boldsymbol{p}$.

Per ogni pixel dell'immagine è analizzata in maniera ordinata l'area circostante, di dimensioni e forma fissate (nel nostro caso la finestra è 9x9) e ogni pixel di quest'area viene confrontato con il pixel generatore. Se il pixel ha una intensità di grigio maggiore viene associato un bit 1 mentre se ha intensità inferiore viene associato il bit 0. L'operatore di confronto è descritto nel seguente modo [4]:

$$\xi(I, p, p') = \begin{cases} 1 & se\ I(p) < I(p') \\ 0 & altrimenti \end{cases}$$

dove I(p)e I(p') sono, rispettivamente, i valori dell'intensità dei pixel p e p'.

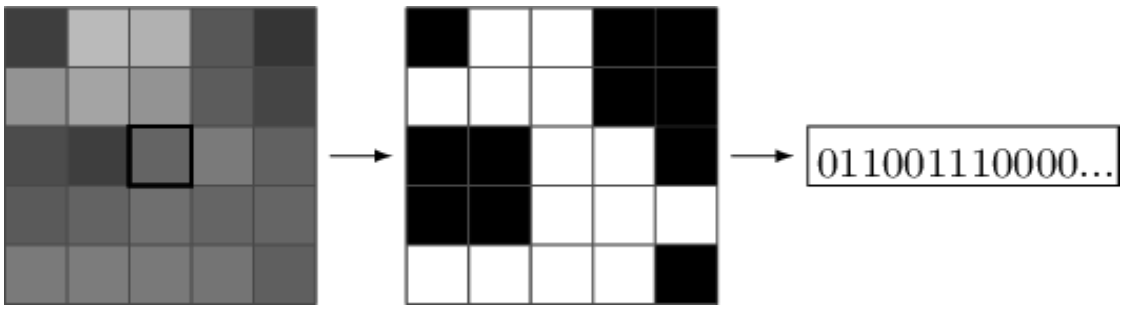


Figure 2.5: Calcolo della trasformazione di Census in un intorno di un punto: l'intorno di un punto viene binarizzato rispetto al valore del punto stesso e da questa sogliatura viene costruita una stringa binaria [4]

Una volta applicata la trasformata a entrambe le immagini, per calcolare il costo di matching tra due punti verrà utilizzata la differenza di Hamming, una semplice operazione tra stringhe di bit che restituisce il numero di bit nei quali esse differiscono.

Rispetto al semplice confronto tra le intensità dei pixel utilizzato nel caso più semplice, questo criterio di somiglianza è preferibile in quanto è:

- Invariante nei confronti di forti distorsioni fotometriche.
- Robusto, tollerante nei confronti di valori anomali dovuti a occlusioni e discontinuità.
- Veloce, essendo le operazioni sono semplici confronti di numeri interi.

2.2.2 Aggregazione dei Costi

La disparità non è più calcolata affidandosi soltanto alla differenza tra un pixel e il presunto pixel omologo, ma è introdotta una funzione di energia più complessa che verrà minimizzata alla ricerca della disparità d migliore [5]:

$$E(d) = E_{data}(d) + E_{smooth}(d)$$

Dove il primo termine indica la funzione dei *costi di matching* da minimizzare lungo l'intera immagine, e il secondo termine serve a penalizzare in modo più o meno marcato eventuali variazioni di disparità tra punti vicini.

Più precisamente, nel caso di SGM, la funzione dei costi globale si può esprimere come:

$$E(D) = \sum_{p} \Big(C\big(p, D_p\big) \Big) + \sum_{q \in N_p} P_1 T\big[\, |D_p - D_q| = 1 \,\big] + \sum_{q \in N_p} P_2 T\big[\, |D_p - D_q| > 1 \,\big]$$

dove [6]:

- *D* è la disparità
- *C* il costo di matching

- P_1, P_2 sono le penalità
- $\boldsymbol{p}$ è un pixel dell'immagine reference
- $\boldsymbol{q}$ è il corrispettivo pixel appartenente al vicinato *Np* nell'immagine target
- D_p, D_q sono le disparità di *p* e *q*, rispettivamente
- N_p è la porzione di immagine attorno a *p*
- *T* è un operatore il cui valore vale 1 se l'argomento è *true* e 0 altrimenti

Questa funzione è calcolata lungo una linea in una particolare direzione (scanline, da cui prende il nome questa euristica, scanline optimization).

Il primo termine indica la somma dei costi di matching per le disparità dell'intera immagine. Il secondo termine aggiunge una penalità costante P1 per tutti i pixel q appartenenti al vicinato di Np di p per i quali si ha una piccola variazione di disparità, qui di 1 pixel.

Il terzo termine aggiunge una penalità maggiore P 2 per tutte le variazioni di disparità maggiori di 1 pixel, nella speranza si preservare le discontinuità dovute alla profondità.

Lo scopo di queste *smoothness penalities* è quello di penalizzare i cambi di disparità tra punti vicini, garantendo così la continuità (smoothness) dell'immagine e permettendo quindi a SGM di adattarsi alle superfici curve o inclinate grazie a P1 (questo perchè negli approcci locali, i pixel appartenenti a superfici inclinate, pur essendo a distanze diverse, spesso sono assegnati alla stessa ipotesi di disparità), ma anche di preservare eventuali discontinuità presenti nella scena grazie alla penalità maggiore P2. Grandi variazioni di disparità sono ritenute valide solo se in corrispondenza di variazioni di profondità. A queste sono spesso associate i bordi appartenenti agli oggetti della scena.

La funzione dei costi appena introdotta viene applicata lungo 4 direzioni differenti, le scanline, in quanto SGM effettua la ricerca di pixel omologhi non su singola scanline, come nel caso base, ma su più percorsi.

Il costo di aggregazione di SGM lungo ogni scanline è quindi determinato ricorsivamente dalla seguente equazione: [3]

$$L_r(\mathbf{p}, d) = C(\boldsymbol{p}, d) + \min\{(L_r(\boldsymbol{p}-\boldsymbol{r}, d), L_r(\boldsymbol{p}-\boldsymbol{r}, d-1) + P_1,\ L_r(\boldsymbol{p}-\boldsymbol{r}, d+1) + P_1, min_i L_r(\boldsymbol{p}-\boldsymbol{r}, i) + P_2)\} -\ min_k L_r(\boldsymbol{p}-\boldsymbol{r}, k)$$

Definito $L_r(\mathbf{p}, d)$ come il costo globale, il primo termine è il costo puntuale relativo al punto $\boldsymbol{p}$ e alla disparità in esame e descrive il costo di matching iniziale; il secondo aggiunge il costo del percorso minimo dei precedenti pixel $\boldsymbol{p}-\boldsymbol{r}$, considerando le

penalità P_1 per i valori con d uguale a $d \pm 1$ e P_2 per i termini con d che va da *dmin* a $d-1$ e da $d+1$ a *dmax* (valori rappresentati dalla variabile *i*); l'ultimo termine rappresenta il minimo costo globale calcolato nel punto precedente $\boldsymbol{p}-\boldsymbol{r}$ e fa sì che il costo di $L_r(\mathbf{p},d)$ non cresca costantemente nell'avanzamento della scanline.

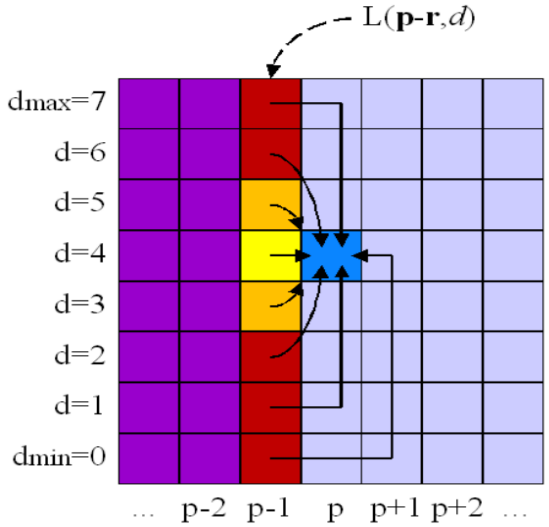


Figure 2.6: i.e. Immaginando di avere 8 possibili disparità, il costo globale nel punto p alla disparità 4 [7]

Una volta valutati i contributi $L_r(\mathbf{p},d)$ relativi ai pixel dell'intera immagine per ogni valore di *d*, si devono considerare i costi globali ottenuti da ogni scanline.

In SGM questo è ottenuto nel seguente modo [3] [7]:

$$S(\boldsymbol{p},d)=\sum_r L_r(\mathbf{p},d)$$

Dove *r* può assumere il valore 4 o 8 a seconda che si sia deciso di utilizzare 4 o 8 differenti scanline, (nel nostro caso 4).

I valori $S(\boldsymbol{p},d)$ vengono quindi memorizzati all'interno di una matrice tridimensionale denominata **DSI** (*Disparity Space Image*)

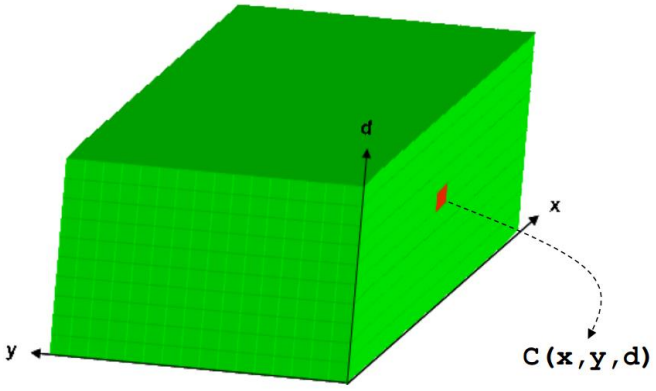


Figure 2.7: Rappresentazione della DSI [5]

Ciascun elemento **C**(x,y,d) del DSI rappresenta il costo della corrispondenza tra il pixel in posizione (x,y) dell'immagine Reference e il pixel in posizione (x+d,y) dell'immagine Target.

2.2.3 Calcolo della Disparità

Una volta raggruppati all'interno della DSI tutti i costi aggregati ottenuti dalle scansioni, si procede cercando il costo dal valore minimo per ogni pixel.

Una volta individuato il minimo per ciascun punto (x,y), al valore della disparità nel punto (x,y) della mappa di disparità risultante verrà assegnato l’indice corrispondente al costo minimo.

CAPITOLO 3

3 IL MACHINE LEARNING

Il Machine Learning basa le sua fondamenta sul problema dell'apprendimento automatico della rappresentazione simbolica di concetti e informazioni relative ad un determinato campo di interesse, sia esso il riconoscimento vocale (Waibel, 1989; Lee 1989), computer vision (i.e. guida autonoma di veicoli su strada (Pomerleau 1989)), classificazione di nuove strutture astronomiche (utilizzato oggi dalla NASA per classificare automaticamente tutti gli oggetti nel Sky Survey), data-mining, e moltissimi altri campi in cui questa disciplina viene applicata e sviluppata.

Per apprendimento automatico si può intendere imparare a svolgere un determinato compito, a prevedere dei risultati, o a classificare dei dati. Tuttavia lo scopo ultimo del Machine Learning, va ben oltre la "semplice" acquisizione di dati e informazioni, ma mira a creare sistemi in grado di apprendere come migliorare la propria conoscenza e il proprio comportamento basandosi sull'esperienza passata, in modo autonomo:

«Un programma apprende da una certa esperienza E se: nel rispetto di una classe di compiti T, con una misura di prestazione P, la prestazione P misurata nello svolgere il compito T è migliorata dall'esperienza E.» [8]

Spesso si ha la necessità di risolvere problemi che risulterebbero di difficile soluzione seguendo un approccio algoritmico tradizionale, studiando cioè un algoritmo che esamini determinate caratteristiche delle immagini o delle altre strutture dati inviate al calcolatore e che implementi una soluzione specifica al problema in questione.

È qui che entra in gioco il Machine Learning: piuttosto che studiare un algoritmo in grado di risolvere il problema direttamente, l'idea alla base dell'apprendimento autonomo è quella di far sì che sia il computer a trovare la soluzione ottimale al problema specifico, partendo però da una conoscenza generale.

Il Machine Learning può essere visto come un tipo di programmazione "per esempi", dal momento che l'unica cosa sulla quale si basa il computer per riuscire a fornire una soluzione al problema è un set più o meno ampio di esempi con relativa soluzione ottimale.

3.1 CLASSIFICAZIONE DEI DIVERSI ALGORITMI

Gli algoritmi di Machine Learning sono tipicamente classificati in tre principali categorie, in base alla natura dell'apprendimento: [9]

- **Apprendimento Supervisionato:** Il calcolatore è preventivamente addestrato tramite un training set, un insieme di attributi relativi ad un elemento in input, ai quali è associato un valore di output chiamato *label* (o etichetta) che indica il risultato di tale elemento.
  Il fine di tale approccio è quello di insegnare al programma una regola generale che mappi l'andamento degli attributi in input a quel determinato risultato(fase di training).
  Una volta individuato il modello migliore possibile, si passa alla fase di testing, in cui l'algoritmo utilizza il modello trovato su una serie di dati di prova(questa volta senza risultato associato) per poter calcolare stime su tali dati. Questo è proprio il tipo di approccio implementato nel framework in questione.
- **Apprendimento Non Supervisionato:** Nessuna etichetta è data all'algoritmo come indicazione su cosa cercare, e l'unico metodo che questo ha a disposizione per trovare un modello generale è quello di confrontare fra di loro i dati in input, cercando analogie e differenze, e riorganizzandoli sulla base di ragionamenti e previsioni sui dati successivi. Questo approccio può essere un obiettivo di per sé (i.e. Per scoprire modelli nascosti nei dati) o un mezzo verso un fine (in funzione dell'apprendimento).
- **Appredimento per rinforzo:** Con questo approccio il programma interagisce con un ambiente dinamico, dal quale riceverà stimoli a seconda della correttezza della scelta compiuta. Il sistema riceverà quindi un premio nel caso la scelta sia corretta, mentre riceverà una penalizzazione nel caso non lo sia. L'apprendimento del modello migliore deriverà dunque dal conseguimento del maggior premio possibile.

Un'altra discriminante è il tipo di output desiderato. Nel caso dell'apprendimento supervisionato possiamo avere diversi tipi di approcci, tra cui:

- **Classificazione:** Nel caso di classificatori, gli input sono suddivisi in due o più classi, e l'algoritmo deve produrre un modello che assegni i nuovi elementi testati ad una o a più di queste classi (*multi-label classification*). Tipicamente implementano in un modello supervisionato.

- **Regressione:** Nel caso della regressione, anch'essa gestita con apprendimento supervisionato, gli output assegnati sono risultati numerici, invece che "descrittori".

Il framework utilizzato per il presente lavoro adotta un approccio del tipo regressivo, associando al training set di dati in input un valore numerico compreso tra 0 e 1 che indica il grado di correttezza stimato per i diversi contributi provenienti dalle varie scanline, e in particolare utilizza un regressore del tipo *RandomForest*, un particolare tipo di classificatore composto da più classificatori semplici detti Alberi Decisionali.

3.2 ALBERI DECISIONALI

Gli alberi decisionali sono utilizzati come modello predittivo che associa all'andamento di un elemento, conclusioni relative al suo valore "concettuale" (l'etichetta). Modelli in cui tale *label* possa assumere un set discreto di valori vengono chiamati *classification trees*. Se invece può assumere valori continui (tipicamente numeri reali) vengono chiamati *regression trees*, come nel nostro caso.

Un albero decisionale può essere immaginato come una struttura gerarchica formata da un insieme di nodi correlati da archi (rami) orientati e "etichettati". La classificazione delle istanze dei dati, avviene ordinandole dalla radice ad una certa foglia, la quale rappresenta il risultato predetto a partire dai valori dei vari attributi e quindi la classificazione del dato in input.
Ogni nodo effettua un test su un attributo, e ogni ramo uscente da un nodo corrisponde ad uno dei possibili valori che l'attributo può assumere. Ciascuna regola di predizione è rappresentata dal cammino (*path*) dal nodo radice (*root*) al nodo foglia. [8]

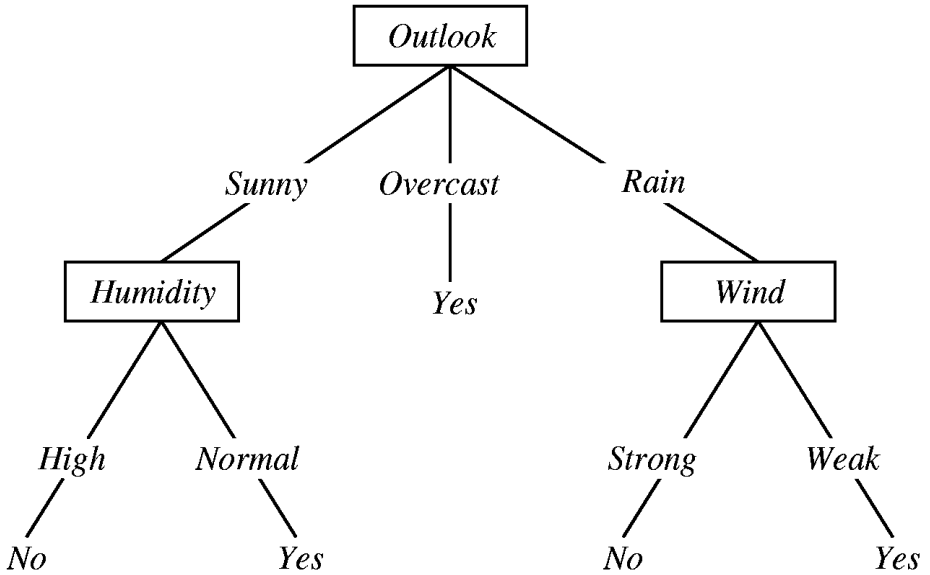


Figure 3.1: Un esempio di albero decisionale per il problema PlayTennis. Classifica Yes/No una data giornata in base a se è adatta a giocare a tennis o meno, considerando varie condizioni metereologiche [8]

3.2.1 Random Forest

Una *Random Forest* è uno speciale classificatore formato da un insieme di classificatori semplici (Alberi Decisionali), rappresentati come vettori random indipendenti e identicamente distribuiti, dove ognuno di essi vota per la classe più popolare in input.

Questo tipo di struttura ha apportato notevoli migliorie nell'accuratezza dell'apprendimento per classificazione/regressione e rientra nella sfera dell'ensemble learning [10].

Ciascun albero presente all'interno di una *Random Forest* è costruito e addestrato a partire da un sottoinsieme casuale dei dati presenti nel training set, gli alberi non utilizzano quindi il set completo, e ad ogni nodo non viene più selezionato l'attributo migliore in assoluto, ma viene scelto l'attributo migliore tra un set di attributi selezionati casualmente.

La casualità è un fattore che entra quindi a far parte della costruzione dei classificatori e ha lo scopo di accrescere la loro diversità e diminuirne così la correlazione.

Il risultato finale restituito dalla *Random Forest* altro non è che la media del risultato numerico restituito dai diversi alberi nel caso di una regressione, o la classe restituita dal maggior numero di alberi nel caso la *Random Forest* sia stata utilizzata per effettuare una classificazione.

CAPITOLO 4

4 MODIFICHE PRECEDENTEMENTE APPORTATE ALL'ALGORITMO SGM TRADIZIONALE

Lo scopo di queste modifiche come accennato precedentemente è stato quello di sfruttare le potenzialità di una Random Forest per migliorare la modulazione dei costi effettuata dall'algoritmo SGM e tentare quindi di ottenere un'accuratezza maggiore dei risultati.

4.1 FRAMEWORK UTILIZZATO

Per ulteriori dettagli rispetto alle informazioni date di seguito, riferirsi a [7].

4.1.1 Random Forest

Il punto cardine attorno al quale si sviluppa il framework utilizzato è la *RandomForest*. Il framework utilizza il componente *CvRTrees* fornito da OpenCV [11], una libreria opensource incentrata sulla Computer Vision e dotata di moduli per il Machine Learning.

4.1.2 Feature utilizzate

È molto importante quando si utilizzano le *Random Forest*, scegliere con attenzione gli attributi da fornire alla *Random Forest*, in quanto è bene che essi siano il più possibile significativi. Le feature, in particolare sono proprietà misurabili rappresentative di un dato elemento in input. Possono essere di qualsivoglia natura, l'importante è che appunto rispecchino e descrivano il dato a cui sono riferite.

Le feature di partenza implementate nel sistema sono la mediana, la deviazione standard, e la deviazione mediana assoluta calcolate su ciascun punto delle mappe di disparità. Scopo del qui presente lavoro sarà quello di andare ad aggiungerne altre a quelle già presenti, con la speranza che dopo la fase di test si dimostrino significative come quanto si attende.

4.1.3 Training

La prima, fondamentale, operazione da compiere quando si lavora con un regressore/classificatore è il training.

Le feature su cui veniva generato il training erano state calcolate su finestre di dimensioni 5x5, 7x7, 9x9, e 11x11 su ciascun punto delle 8 mappe di disparità

ottenibili prendendo in esame i contributi provenienti da una singola scanline per volta. C'è da sottolineare come il framework sia stato impostato per l'utilizzo di 8 scanline, ma il presente lavoro abbia modificato questo numero ridimensionandolo a 4 per motivi di potenza di calcolo al fine di ridurne il carico computazionale, mantenendo solo le scanline verticali e orizzontali in entrambi i versi.

A questo punto vi è la generazione di un file di training in formato csv composto da tanti record quanti sono i pixel validi delle 64 mappe ottenute.

Ogni record era composto dalla sequenza dei valori delle 12 feature in quel punto e da un risultato numerico che può assumere i valori 1 o 0 a seconda rispettivamente che l'intensità del pixel analizzato si discosti di 3 o meno da quella del corrispondente pixel della *ground truth* (ovvero una mappa di disparità relativa alla coppia stereo in cui le profondità sono sicuramente corrette, essendo state rilevate tramite scansioni laser) o più di 3.

4.2 MODIFICHE APPORTATE A SGM

La prima parte di SGM resterà immutata. Le differenze tra pixel saranno quindi calcolate tramite la differenza di Hamming applicata alle trasformate Census delle coppie di immagini Reference e Target come accade nell'implementazione di SGM scelta come base di partenza. Inoltre, anche i costi lungo le singole scanline saranno trovati nella maniera originaria. Ciò che è cambiato è il modo in cui vengono aggregati i costi calcolati lungo le 8 differenti scanline.

Anziché sommarli e memorizzare i risultati all'interno di una DSI come eseguito dall'approccio tradizionale, infatti, sono stati sperimentati quattro nuovi approcci:

1. **Scelta del contributo proveniente dalla scanline migliore**
2. **Somma dei contributi delle 4 scanline più affidabili**
3. **Somma pesata dei contributi delle 4 scanline più affidabili**
4. **Somma pesata dei contributi delle 8 scanline**

Dai risultati sperimentali è in seguito risultato che la somma pesata degli otto contributi sia l'unica a dare risultati migliorativi rispetto a SGM standard, dando l'errore percentuale medio minore.

Il presente lavoro, alla luce di tali risultati, sarà quindi sviluppato effettuando confronti solo tramite la **somma pesata dei contributi della totalità delle scanline**, nel nostro caso 4.

CAPITOLO 5

5 MODIFICHE PROPOSTE AL FRAMEWORK

Utilizzando un framework dove l'analisi delle immagini avviene in maniera estremamente direzionale (tramite scanline), viene naturale chiedersi se non possa essere di giovamento al sistema un'impostazione completamente direzionale.

Per questo sono state attuate varie modifiche, descritte di seguito, volte a dare maggiore considerazione e importanza alle singole scanline, essendo esse potenziali portatrici di informazioni aggiuntive, se separate le une dalle altre.

Il presente lavoro ruota attorno a questo: dimostrare o confutare tali miglioramenti.

5.1 IMPOSTAZIONE DIREZIONALE DELL'ARCHITETTURA DEL FRAMEWORK

Il framework così per com'era stato ideato, si basava sul calcolo delle feature su patch (le finestre di supporto) di varie dimensioni, ma sempre dalla forma quadrata. Per via di tale geometria, non vi era alcun tipo di diversificazione fra le informazioni acquisite dal calcolo degli attributi nel vicinato del pixel in esame, lungo le varie scanline.

La prima modifica apportata è stata quindi l'impostazione dell'intero sistema su patch rettangolare.
Introducendo finestre rettangolari, ci si deve porre il problema di come esse debbano essere orientate rispetto alla scanline.

Nel caso di patch quadrate il problema non si poneva: non esisteva il concetto di verticale o orizzontale, e ortogonale o parallelo.

Si è quindi deciso di aggiungere una semplice funzione di utilità ***patch_size()*** che, dato un intero ***base*** utilizzato come dimensione base, e il numero della scanline corrente, restituisca un oggetto ***Size(int width, int height)*** con le seguenti dimensioni nel caso di patch:

- Ortogonali:
  - scanline orizzontali: larghezza pari a *base* e altezza pari a *base·N*
  - scanline verticali: larghezza pari a *base·N* e altezza pari a *base*
- Parallele:
  - scanline orizzontali: larghezza pari a *base·N* e altezza pari a *base*

- scanline verticali: larghezza pari a *base* e altezza pari a *base·N*

con N, moltiplicatore scelto a seconda di quanto si vuole grande la patch.

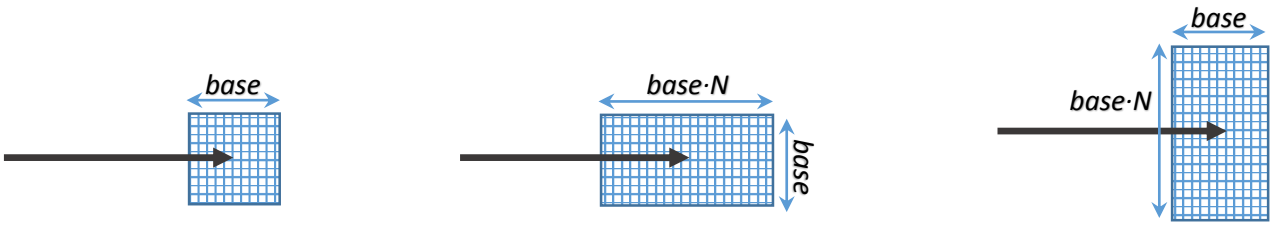


Figura 5.1: Disegno dei vari tipi di patch. Quadrata, parallela e ortogonale.

In seguito si vedrà come sono state ricercate la forma (quadrata o rettangolare), l'orientamento (ortogonale o parallelo) e le dimensioni (*base* e *N*) ottimali della finestra di supporto in base anche (e soprattutto) ai risultati sperimentali ottenuti.

Un altro passo verso un'architettura interamente direzionale è stato compiuto in fase di creazione della foresta. Nell'approccio iniziale la foresta era generata a partire da un unico file di training scritto ad ogni scansione delle scanline. L'approccio proposto e sperimentato fa sì che siano invece generati 4 differenti file di training, ognuno specifico della scanline corrente. Avremo quindi alla fine del processo di training del sistema 4 file *training_i.train* (con i = 0,1,2,3 indicativo della scanline) ognuno contenente un quarto dei training sample (il totale sarebbe il n° dei pixel validi delle immagini su cui vengono calcolate le feature moltiplicato per il n° delle scanline). Questa suddivisione dei training sample fa sì che si possano creare 4 differenti foreste, addestrate "direzionalmente", *forest_i.xml*, con i = 0,1,2,3.

Tali foreste saranno poi utilizzate in fase di testing e previsione delle mappe di disparità risultanti e verrà chiesto loro di fornire previsioni sulla base della scanline a cui corrispondono.

5.2 FEATURE PROPOSTE

Come accennato precedentemente le feature di partenza implementate nel framework erano 12: mediana, deviazione standard e deviazione mediana assoluta ognuna calcolata su 4 finestre quadrate dalle diverse dimensioni sulle mappe di disparità.

Per calcolare la mediana OpenCv mette a disposizione il filtro ***medianBlur(),*** che tra i vari parametri della signature prende in ingresso un intero *ksize*, rappresentativo della dimensione del lato della patch di supporto su cui calcolare appunto la mediana. La finestra può quindi essere solo quadrata. Per questo si è dovuto creare una funzione *ad hoc* per calcolare la mediana su patch rettangolare:

void medianBlur_rettangolare(Mat InputMat, Mat & OutputMat, Size patch_size)

che invece di un semplice intero prenda in ingresso un oggetto *Size* generato da ***patch_size(),*** che fornisce la finestra rettangolare.

Le uniche informazioni estrapolate sulle scene in oggetto, erano quindi esclusivamente provenienti dalle mappe di disparità calcolate da SGM a partire dalla coppia stereo in esame, composta da left e right image.

Nessun tipo di analisi veniva effettuato sulle immagini di partenza. Si è quindi pensato di introdurre feature che lavorassero direttamente sull'immagine Reference (Left) della stereo pair, in modo da poter ricavare informazioni potenzialmente significative anche da loro.

La prima modifica apportata alle feature è stata quindi calcolare mediana, deviazione standard e deviazione mediana assoluta (MAD) anche sulle Left Images. Nel file csv di training per ora, non avremo più solo 12 feature, ma bensì il doppio. Il risultato etichetta, invece, continua ad essere calcolato verificando la condizione di cui sopra, solo fra ground truth e mappa di disparità.

Un'ulteriore semplice modifica è stata quella di sostituire la MAD con l'opposto del valore assoluto della differenza tra l'immagine in input (sia mappe di disparità che left image) e la mediana calcolata su essa come proposto in [12], dimostrandosi efficace per lo scopo.

A questo punto l'analisi della Left Image continua introducendo due nuove feature: la *distanza da un edge direzionale* e la *distintività*.

5.2.1 Feature distanza da un edge direzionale

Lo scopo della feature distanza da un edge direzionale è quello di calcolare la distanza tra ogni pixel e il primo *edge* (bordo) a esso più vicino, cercato in senso opposto alla scanline.

Al fine di calcolare tale feature, è stata creata una funzione di supporto che calcolasse appunto gli *egde direzionali* delle scene in esame.

Gli *edge*, in termini consoni alla compuer vision, sono quei punti in cui si ha una significativa variazione di intensità. Un metodo matematico per esprimere i cambiamenti, è rappresentato dall'uso dei gradienti, in particolare useremo gradienti applicati alla funzione intensità dei pixel; si ottiene così la funzione gradiente. Un gradiente elevato indica un'elevata variazione, e quindi la presenza di un edge.

Quindi in generale, possiamo assumere che un metodo per individuare i bordi di un'immagine sia quello di localizzare i pixel con gradiente più elevato in un vicinato di punti, o generalizzando, più elevato di una certa soglia. [13]

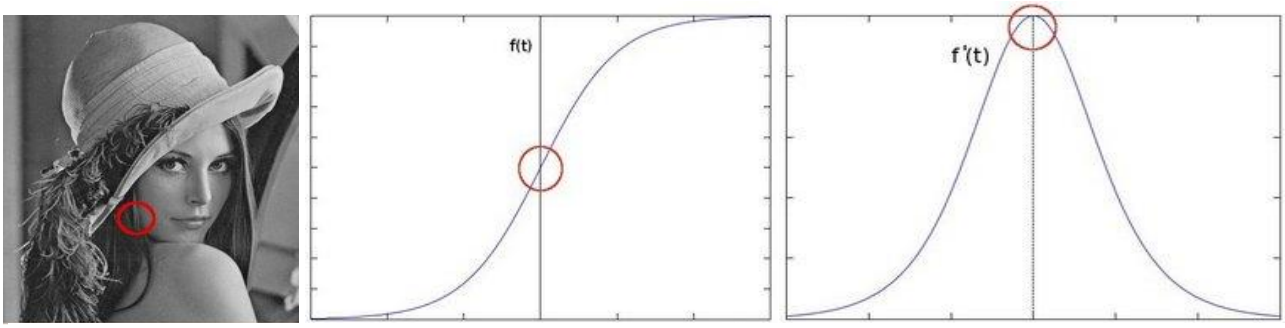


Figure 5.2: Un edge è individuato dal "salto" in intensità, come mostrato nel primo grafico. Questo salto può essere visto ancora più facilmente calcolando la derivata prima di tale intensità, mostrata nel secondo grafico [13]

Parlando più in dettaglio della funzione:

int directional_edge(IplImage *Input, Mat &GradientMat, int Scanline)

essa in base alla scanline datale in ingresso, calcolerà la rispettiva derivata parziale individuando così gli edge direzionali.

Per fare un esempio concreto, durante l'avanzamento della scanline LEFT_RIGHT, essendo questa orizzontale, sarà calcolato il gradiente sullo spostamento in dx evidenziando i bordi verticali.

Figure 5.3: Immagine risultante del gradiente su dx

A questo punto una seconda funzione:

int distance_to_directional_edge(IplImage *Input, Mat &DistanceMat, int Scanline)

ciclerà su ogni pixel appartenente a ***GradientMat***, generata sfruttando la funzione ***directional_edge()***, e calcolerà la distanza tra il pixel corrente e il primo bordo incontrato scorrendo l'immagine da destra a sinistra (nel caso di scanline LEFT_RIGHT).

Figure 5.4: Mappa delle distanze relativamente alla scanline LEFT_RIGHT

Figure 5.5: Mappa delle distanze relativamente alla scanline RIGHT LEFT

Figure 5.6: Immagine risultante del gradiente su dy

Figure 5.7: Mappa delle distanze relativamente alla scanline DOWN_UP

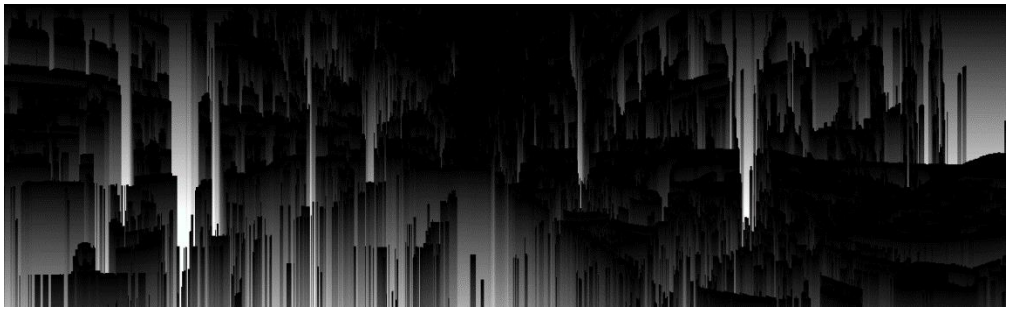


Figure 5.8: Mappa delle distanze relativamente alla scanline UP_DOWN

5.2.2 Feature edge direzionale

Si è in seguito deciso di sfruttare la funzione ***directional_edge()*** anche come feature aggiuntiva da calcolare però solo sulle mappe di disparità, in quanto potenzialmente molto significativa nel loro caso, mentre calcolata sulle Left Image non lo sarebbe affatto. Questo perché il risultato sulle prime evidenzia elevate discontinuità di depth, mentre sulle seconde evidenzia semplicemente gli edge relativi ai soggetti della scena, che potrebbero però trovarsi anche alla stessa profondità, informazione non particolarmente significativa.

Figura 5.9: Mappa di disparità su cui calcoleremo la feature edge_direzionale

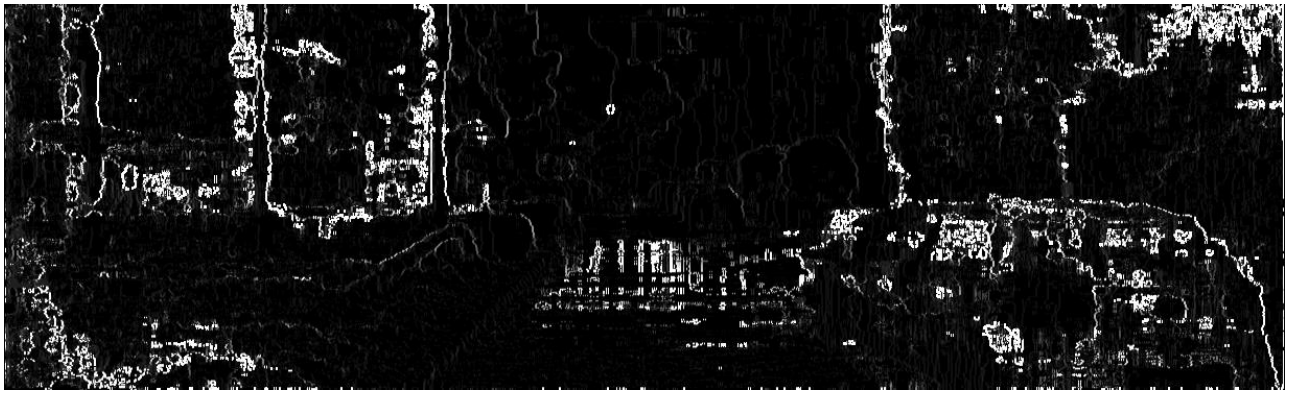


Figura 5.10: Mappa risultante, che evidenzia le discontinuità di depth

5.2.3 Feature distinctiveness

Per parlare della feature *distinctiveness*, sarà bene introdurre il concetto di distintività tra pixel, utilizzata in genere come stima di confidenza.

Il fine di tale stima basata sulla distintività, è quello di gestire nel migliore dei modi le ambiguità nelle scene in esame, dal momento che persino per punti significativi (come ad esempio punti appartenenti a bordi o ad angoli), può risultare difficile trovare una corrispondenza nel caso di pattern ripetitivi.

Grazie alla nozione di distintività (DTS), è meno probabile che avvenga erronea corrispondenza fra punti distintivi fra le immagini di Reference e Target.

La distintività di un punto appartenente a un'immagine è definita come la distanza (intesa come differenza in valore assoluto) percentuale fra questo e il punto più simile ad esso all'interno della finestra in cui si sta effettuando la ricerca, nell'immagine Reference.

$$d_{min} - d_{max} \leq d_s \leq d_{max} - d_{min}$$

dove d_s è la finestra di ricerca della disparità e d_{min} e d_{max} sono rispettivamente il minimo e il massimo della disparità.

A questo punto la distintività di un pixel è data da:

$$C_{DTS}(x,y) = min_{d \in d_s; d \neq 0}\{c_{LL}(x,y,d)\}$$

in cui c_{LL} è il costo della corrispondenza fra pixel giacenti sulla stessa scanline all'interno della stessa immagine (Left Image). È da puntualizzare come la DTS sia proprietà di una singola immagine, dal momento che l'immagine Target non viene presa in considerazione [14].

Si è quindi deciso di inserire la DTS come attributo sulla Left Image. Tuttavia si è adottata una finestra di ricerca lievemente differente: si effettua la ricerca solo fra i punti che precedono il pixel in esame, quindi solo in verso opposto all'avanzamento della scanline.

Avremo quindi:

$$| d_s | \leq range$$

Con $d_s \begin{cases} < 0, & se\ scanline\ LEFT \rightarrow RIGHT\ e\ UP \rightarrow DOWN \\ > 0, & se\ scanline\ RIGHT \rightarrow LEFT\ e\ DOWN \rightarrow UP \end{cases}$

e $d_s \neq 0$ sempre, altrimenti il pixel in esame verrebbe confrontato con sé stesso, vanificando l'utilità della DTS.

È stato possibile ridimensionare la finestra dal momento che non si sta utilizzando la funzione distintività come stima di confidenza di matching, ma bensì come feature. Quindi in questo caso essendo interessati esclusivamente alla "storia passata" del pixel in esame possiamo ridimensionarla come più conviene.

Inoltre la ricerca di distintività ci fornisce informazioni su possibili discontinuità nella *depth*: una bassa distintività lungo la scanline indica regioni uniformi, in cui difficilmente ci aspetteremmo fenomeni di streaking; al contrario, una distintività più elevata si ha in presenza di discontinuità, che possono rappresentare (non necessariamente) discontinuità nella depth e dare luogo a streaking.

5.3 TRAINING

La feature in definitiva selezionate per la fase di *training* e *testing* sono quindi in totale 27: la mediana, la deviazione standard, la deviazione assoluta mediana sia su mappe di disparità che su Left Image calcolate su finestre di 4 dimensioni diverse (si veda in seguito quali), la distanza dal successivo edge direzionale e la distintività su Left Image e infine gli edge direzionali sulle mappe di disparità.

Le coppie di immagini scelte per la fase di training sono la 43, 71, 82, 87, 94, 120, 122, 180 del training set di KITTI come in [7].

È stato scelto il dataset KITTI in quanto dataset creato a partire da scenari reali urbani, raffiguranti strade e luoghi all'aperto (scene quindi particolarmente difficili da analizzare per via di scompensi fotometrici dovuti all'esposizione al sole, aree prive di texture come il cielo, pattern ripetitivi, riflessi, trasparenze, etc), e non da scenari creati *ad hoc* in laboratorio.

A parte il numero e il set di feature, il processo di training rimane invariato rispetto a quanto spiegato nel capitolo precedente.

CAPITOLO 6

6 RISULTATI SPERIMENTALI

Tutti i test preliminari mirati all'individuazione della combinazione migliore, sono stati effettuati sul dataset Middlebury 2014 (previo training su KITTI) in quanto dataset che meglio si presta a test frequenti per via dei tempi di calcolo notevolmente inferiori rispetto a KITTI (un test su Middlebury dura circa un'ora, mentre su KITTI un test sul set completo delle stereo pair può impegare anche molto più di 12 ore). Questo a causa della dimensione e della quantità delle immagini del dataset KITTI.

Nonostante i tempi di elaborazione più elevati, i test sul dataset KITTI sono anche quelli più significativi, per cui una volta individuato il caso migliore tramite Middlebury si è passato poi alla fase di testing definitiva sulle immagini di KITTI.

6.1. RICERCA DELLA DIMENSIONE E FORMA OTTIMALE DELLE FINESTRE DI SUPPORTO

TEST	PATCH	DIMENSIONE PATCH (per scanline orizzontali)	BASI	*range* (DTS)
1°	ortogonali	(base, base·3)	5,7,9,11	11
2°	ortogonali	(base, base·2)	5,7,9,11	11
3°	ortogonali	(base, base·2)	3,5,7,9	9
4°	ortogonali	(base, base·2)	3,5,7,9	20
4°bis	parallele	(base·2, base)	3,5,7,9	20

Table 6.1: Caratteristiche principali dei test effettuati

La tabella riporta alcuni dei test (quelli più significativi) che sono stati effettuati per trovare dimensione/forma delle patch, e la dimensione del *range* da esaminare nella funzione DTS, ottimali.

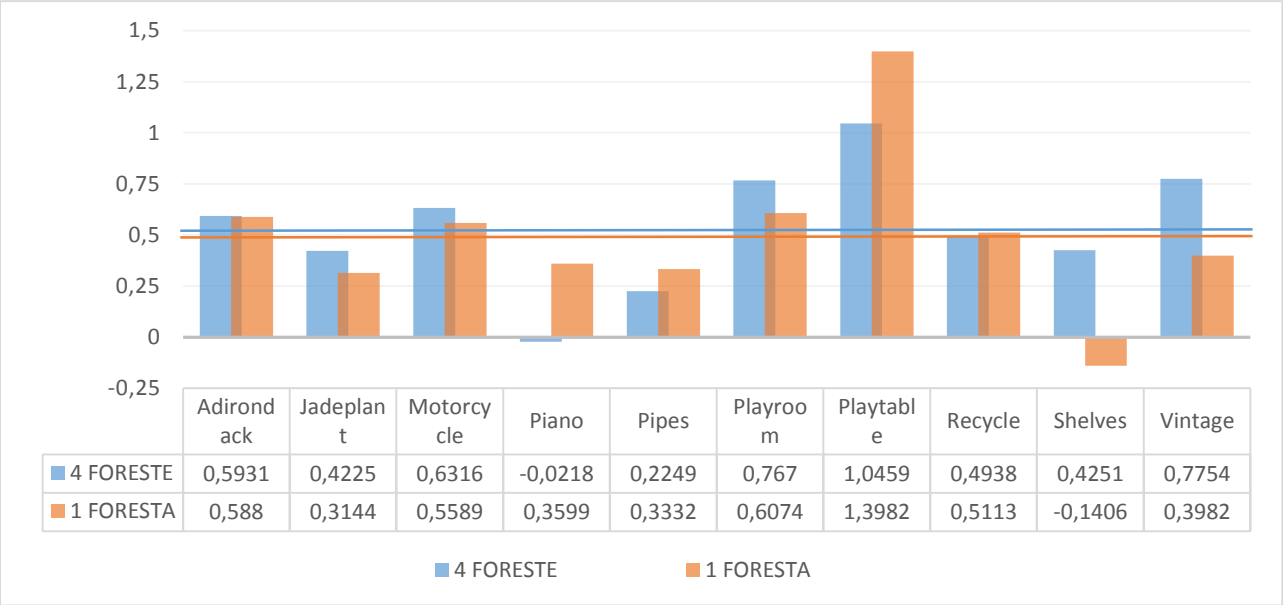
Si vuole puntualizzare che ogni test è stato compiuto utilizzando sia l'approccio con quattro foreste direzionali che a una singola foresta non direzionale. In questo modo è stato più facile e intuitivo verificare se effettivamente la creazione di quattro foreste apportasse benefici.

Per quanto riguarda la scelta di *range* si è inizialmente deciso che avesse la dimensione del lato parallelo alla scanline corrente, il più corto quindi nel caso di patch ortogonali; il più lungo nel caso invece fossero parallele. Si è però in seguito notato che a un suo aumento (purchè non eccessivo), corrispondeva un lieve miglioramento.

6.1.1 Risultati sperimentali dei test preliminari sul dataset Middlebury

Vediamo i test preliminari più nel dettaglio:

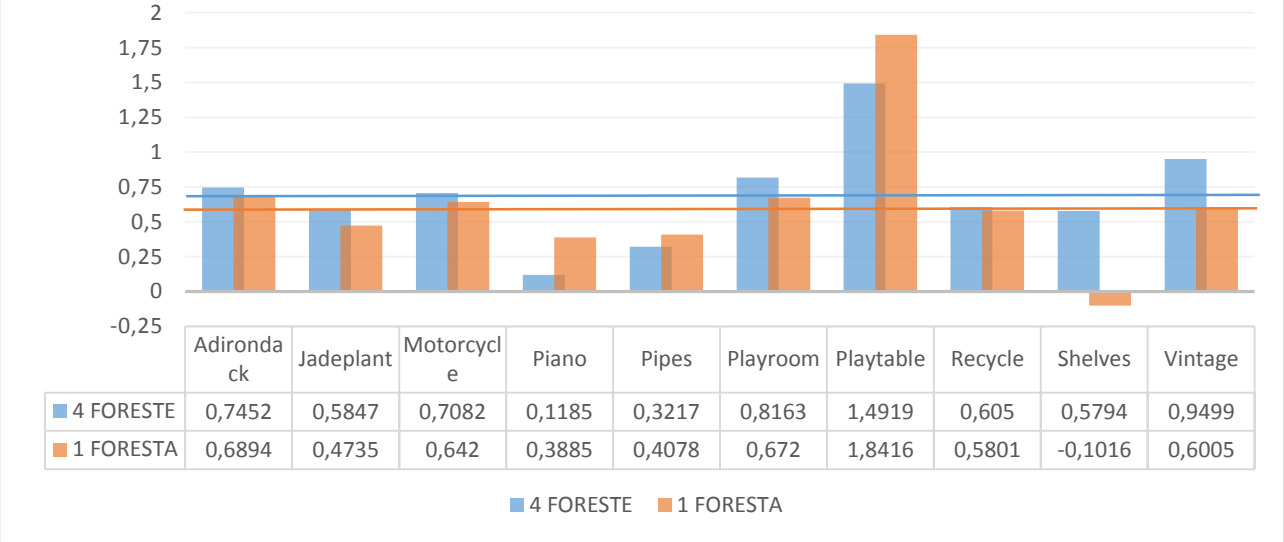
1°: Come primo tentativo si è voluto mantenere le dimensioni delle finestre di base così come già impostate nel sistema quindi 5, 7, 9 e 11 e moltiplicando per 3 per ottenere il lato lungo, volendo una finestra rettangolare.



	Adirondack	Jadeplant	Motorcycle	Piano	Pipes	Playroom	Playtable	Recycle	Shelves	Vintage
4 FORESTE	0,5931	0,4225	0,6316	-0,0218	0,2249	0,767	1,0459	0,4938	0,4251	0,7754
1 FORESTA	0,588	0,3144	0,5589	0,3599	0,3332	0,6074	1,3982	0,5113	-0,1406	0,3982

Graph 6.1: Differenza dell'errore percentuale rispetto a SGM nel Test #1. Sulle ascisse sono riportate le immagini del dataset

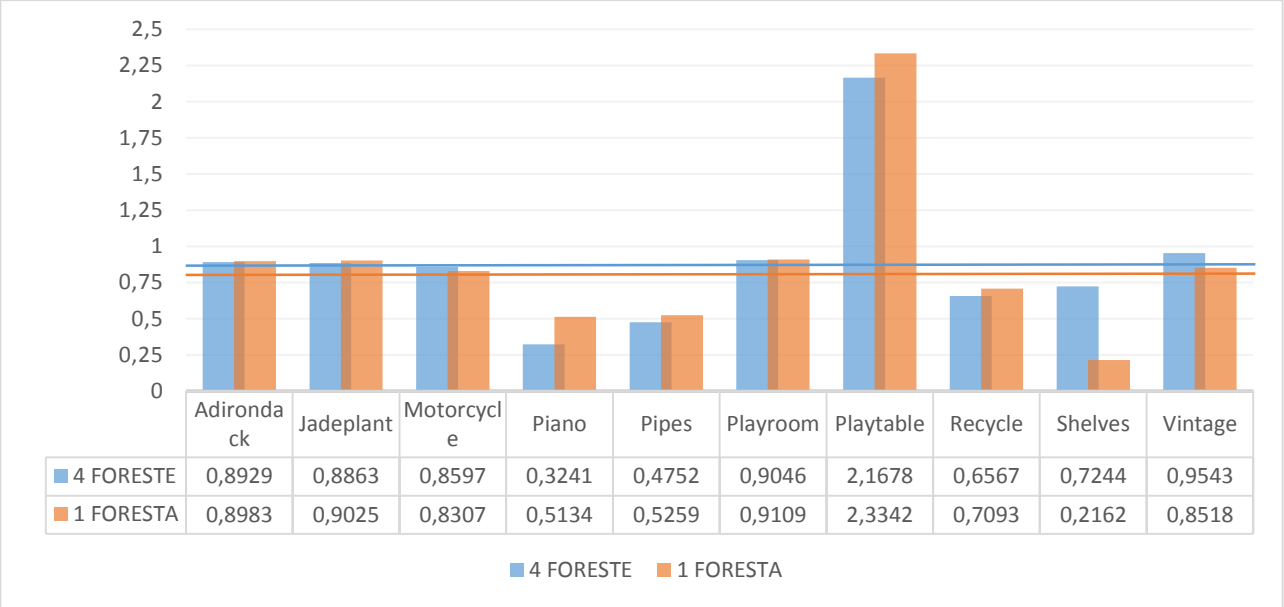
2°: Si è poi passato a ridurre le dimensioni delle finestre, passando a un moltiplicatore più piccolo, da 3 a 2.



	Adirondack	Jadeplant	Motorcycle	Piano	Pipes	Playroom	Playtable	Recycle	Shelves	Vintage
4 FORESTE	0,7452	0,5847	0,7082	0,1185	0,3217	0,8163	1,4919	0,605	0,5794	0,9499
1 FORESTA	0,6894	0,4735	0,642	0,3885	0,4078	0,672	1,8416	0,5801	-0,1016	0,6005

Graph 6.2: Differenza dell'errore percentuale rispetto a SGM nel Test #2. Sulle ascisse sono riportate le immagini del dataset

3°: Notando un miglioramento si è continuato a ridurre la dimensione delle finestre, questa volta cambiando proprio le basi di partenza: ora sono 3,5,7,9, e sono il minimo applicabile. A questo punto il parametro ***range*** vale 9.

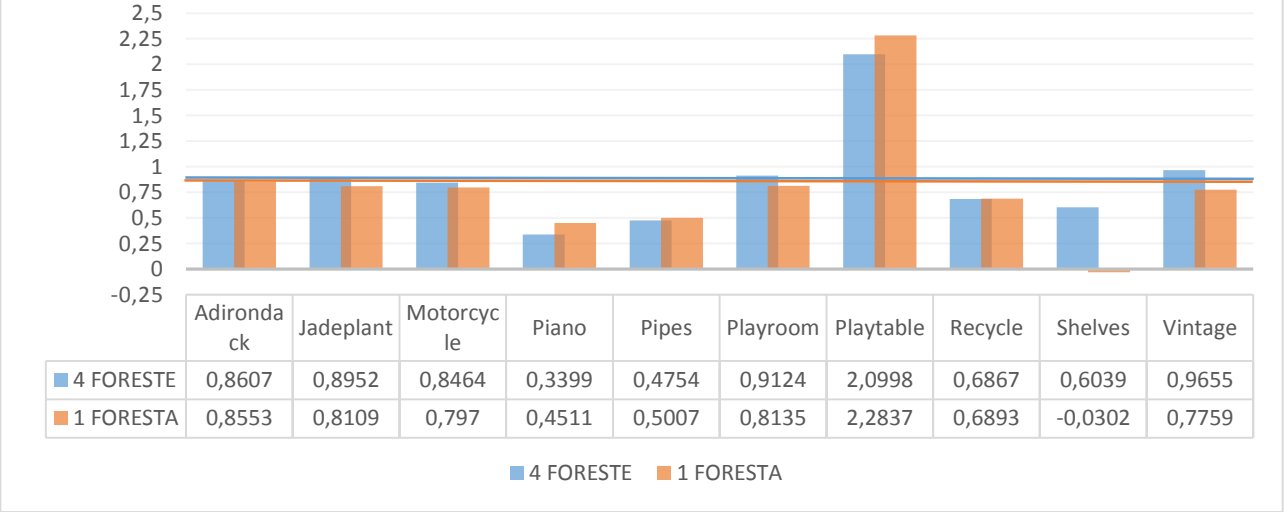


	Adirondack	Jadeplant	Motorcycle	Piano	Pipes	Playroom	Playtable	Recycle	Shelves	Vintage
4 FORESTE	0,8929	0,8863	0,8597	0,3241	0,4752	0,9046	2,1678	0,6567	0,7244	0,9543
1 FORESTA	0,8983	0,9025	0,8307	0,5134	0,5259	0,9109	2,3342	0,7093	0,2162	0,8518

Graph 6.3: Differenza dell'errore percentuale rispetto a SGM nel Test #3. Sulle ascisse sono riportate le immagini del dataset

Arrivati a una dimensione ottimale della patch si è tentato quindi di ridimensionare il parametro *range* della feature Distintività, per vedere se ci potessero ancora essere miglioramenti.

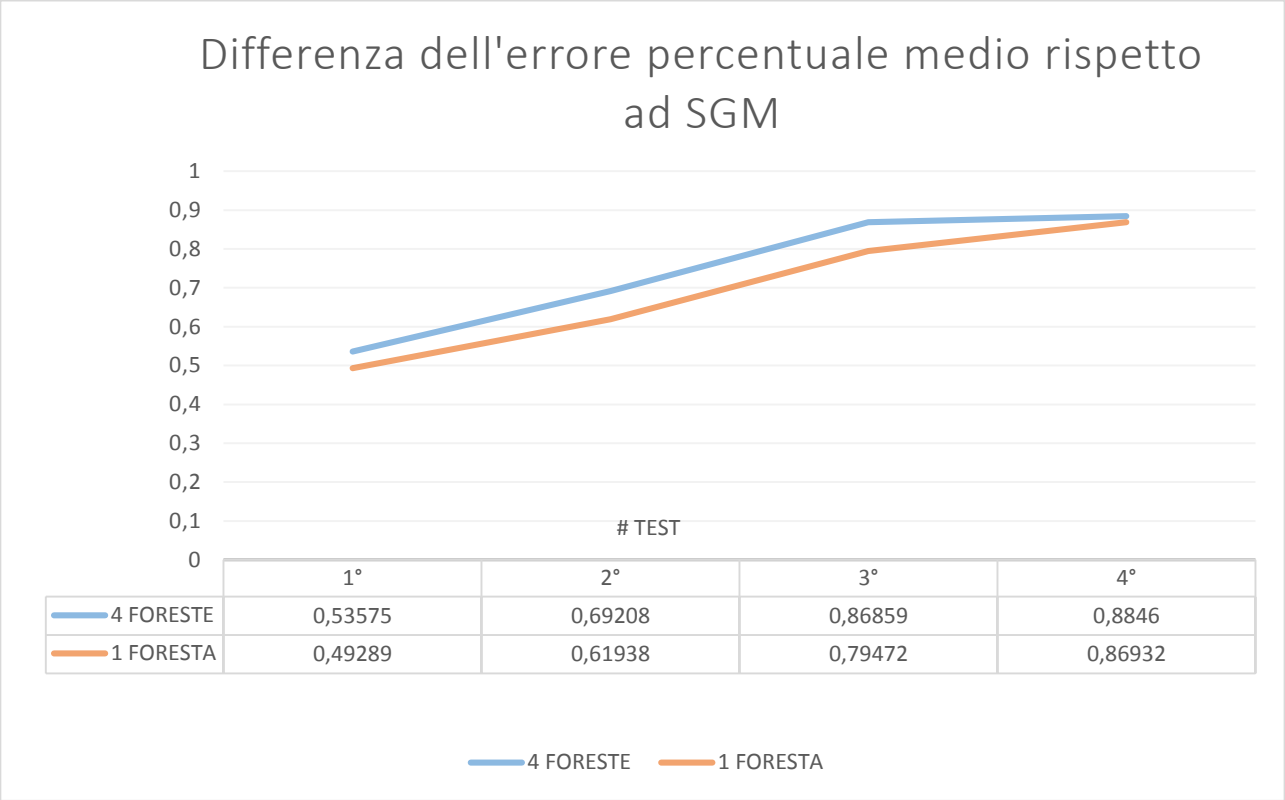
4°: Dopo un test peggiorativo con *range* settato a 50, si è abbassato tale valore fino a ottenere risultati migliorativi, con *range* settato a 20.



	Adirondack	Jadeplant	Motorcycle	Piano	Pipes	Playroom	Playtable	Recycle	Shelves	Vintage
4 FORESTE	0,8607	0,8952	0,8464	0,3399	0,4754	0,9124	2,0998	0,6867	0,6039	0,9655
1 FORESTA	0,8553	0,8109	0,797	0,4511	0,5007	0,8135	2,2837	0,6893	-0,0302	0,7759

Graph 6.4: Differenza dell'errore percentuale rispetto a SGM nel Test #4. Sulle ascisse sono riportate le immagini del dataset

Volendo a questo punto generalizzare i risultati ottenuti:

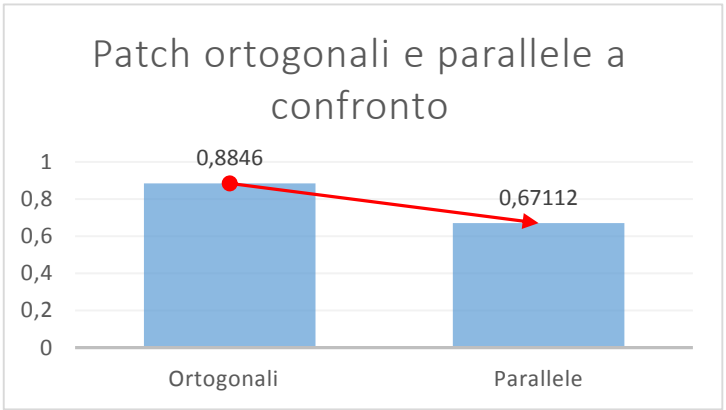


Graph 6.5: Andamento della differenza di errore percentuale medio rispetto ad SGM nei vari test

Il grafico 6.1 evidenzia la differenza, in media, tra l'errore ottenuto utilizzando SGM standard e l'errore ottenuto con le modifiche apportate dai vari test.

In generale, si evince che diminuendo la dimensione delle patch (il quarto test è quello con le patch più piccole) si ha una diminuzione nell'errore e quindi un miglioramento.
Tuttavia l'informazione più interessante è il miglioramento globale introdotto dalle quattro foreste direzionali, le quali danno sempre i risultati medi migliori.

Un ultimo test è stato fatto a partire dal caso migliore finora individuato: il Test #4 con approccio a 4 foreste; e è stato effettuato utilizzando patch parallele alla scanline, con lo scopo di verificare l'effettiva superiorità delle patch ortogonali.



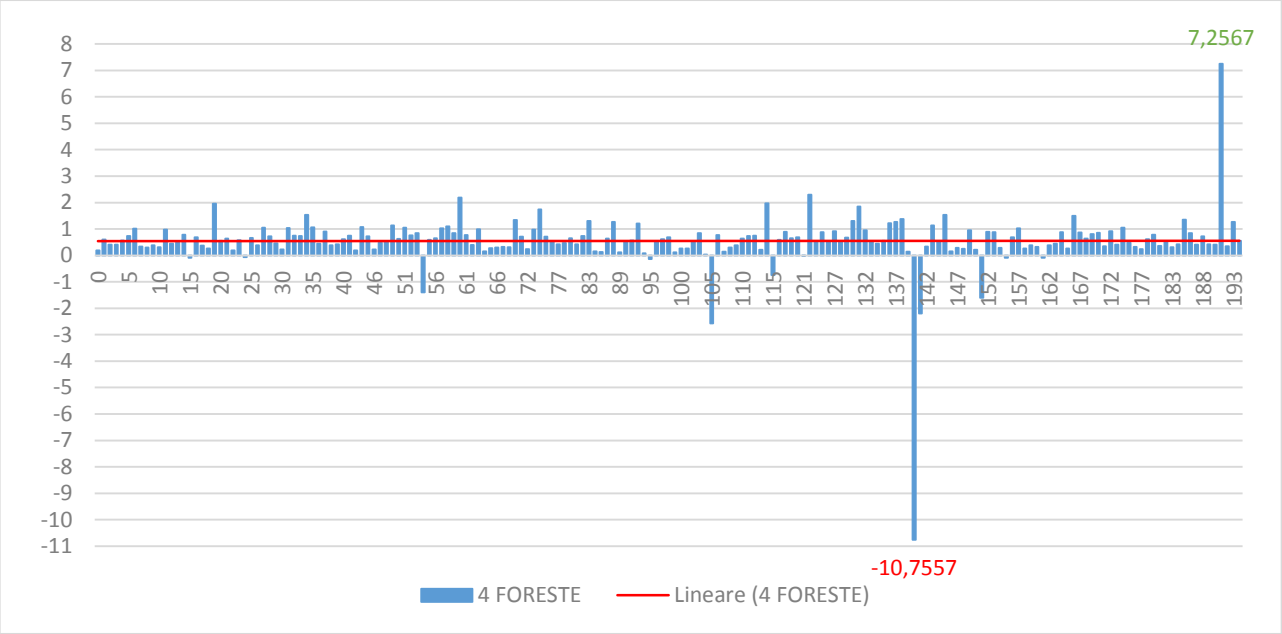
Graph 6.6: confronto tra patch ortogonali e parallele a partire dal caso migliore

Come si evince dal grafico 6.6 questa modifica ha apportato un peggioramento, facendoci così scartare tale "combinazione".

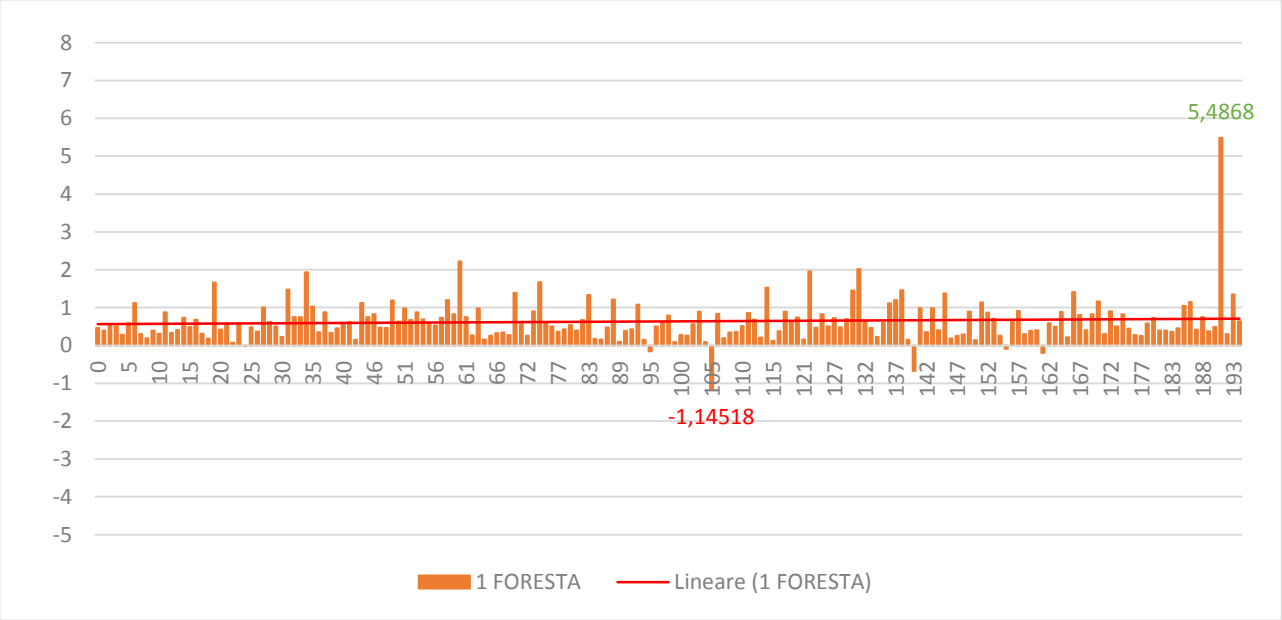
6.2 RISULTATI SPERIMENTALI OTTENUTI SUL DATASET KITTI

Una volta individuata la combinazione di caratteristiche migliore, si è quindi passato alla fase di testing sulle 193 stereo pair di KITTI.

6.2.1 Differenza dell'errore percentuale rispetto a SGM



*Graph 6.7: Differenza media dell'errore percentuale rispetto a SGM per sistema a 4 foreste, **visualizzato per ogni coppia di immagini del training set di KITTI**. Valori positivi indicano un miglioramento*

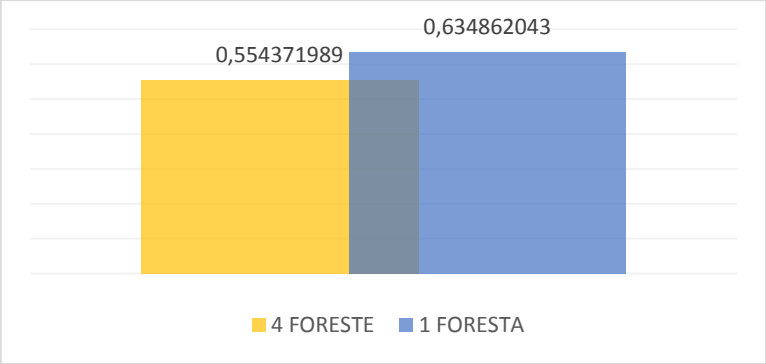


*Graph 6.8: Differenza media dell'errore percentuale rispetto a SGM per sistema a singola foresta. **visualizzato per ogni coppia di immagini del training set di KITTI**. Valori positivi indicano un miglioramento*

I grafici 6.7 e 6.8 evidenziano la differenza tra l'errore ottenuto utilizzando SGM e l'errore ottenuto con la combinazione del test migliore (il quarto), rispettivamente con il sistema impostato a 4 foreste e a singola foresta.

Come si può notare, a parte sporadici casi (i.e. il più evidente è il caso dell'immagine n°140 nel sistema a 4 foreste) la tendenza dei risultati è quella di apportare miglioramento a SGM standard.

Mettendo ora a paragone i risultati medi dei due sistemi vedremo quale è stato il più efficace.



Graph 6.9: Differenza dell'errore medio tra SGM e le modifiche apportate

Contrariamente a quanto ci si aspettava, visti i risultati dei test sul dataset Middlebury, è stato il sistema a singola foresta a ottenere i risultati migliori, sebbene la differenza con il corrispettivo a 4 foreste non sia elevata. Questo potrebbe essere dovuto al fatto che i due dataset sono profondamente diversi sotto molti punti di vista, quali le scene raffigurate, le caratteristiche tecniche delle immagini, la massima disparità ricercata, le ground truth, etc. Per fare un semplice esempio, Middlebury si concentra esclusivamente su scene *indoor* appositamente ricreate in laboratorio, difficilmente prive di texture, mentre KITTI su scene *outdoor* urbane e quindi con esposizione fotometrica estremamente diversa, aree prive di texture o pattern ripetitivi, immagini sicuramente dalla più difficile analisi. Questa differenza nei due dataset è la causa prevalente dell'incongruenza dei risultati.

6.2.2 Riduzione percentuale dell'errore del caso migliore su KITTI

Concentriamoci ora sul margine di miglioramento più elevato: 0.634862043, ottenuto grazie alla modifica delle dimensioni delle patch e l'introduzione delle nuove feature, su sistema a singola foresta. Può risultare un margine di miglioramento ridotto, ma bisogna ricordare che si tratta di miglioramenti relativi ad un algoritmo (SGM) già estremamente accurato nella sua versione standard, per cui "a un margine di miglioramento apparentemente minimo in termini assoluti corrisponde in realtà un incremento prestazionale significativo in termini relativi" [7]. Per enfatizzare quanto detto calcoleremo la riduzione percentuale dell'errore medio relativo. A una

differenza media dell'errore percentuale pari a 0.634862043 corrisponde una riduzione del 7.19% dell'errore relativo commesso rispetto a SGM [15].

6.3 CONFRONTO QUALITATIVO TRAMITE MAPPE DI DISPARITÀ

Al fine di dare un'idea più concreta e chiara di ciò che è stato fatto si procede con un confronto diretto fra le mappe di disparità calcolate su KITTI con SGM standard e il sistema proposto nel caso migliore, sia impostato con singola foresta, che con 4 foreste direzionali.

6.3.1 Sistema nel caso migliore in assoluto: patch rettangolari e ortogonali e creazione di singola foresta di training

6.3.1.1 Risultato migliore

In questo scenario, la stereo pair che ha maggiormente permesso di ridurre l'errore percentuale medio di SGM standard è stata la coppia n°191 del training set di KITTI.

Figure 6.1: Immagine Left della coppia n° 191

Figure 6.2: Mappa di disparità, ottenuta dall'algoritmo SGM originale

Figure 6.3: Mappa di disparità, ottenuta dal sistema modificato

Dal confronto delle Figure 6.2 e 6.3 appare evidente come le modifiche apportate al sistema abbiano migliorato la stima di disparità calcolata da SGM, eliminando un grosso errore (cerchiato di rosso nella Figura 6.2) al centro della strada, dovuto probabilmente al forte riflesso della luce.

In questo caso, il sistema proposto è riuscito a migliorare SGM del 27.08%, commettendo un errore del 14.7774% contro un errore del 20.2642% commesso da SGM [15].

6.3.1.2 Risultato peggiore

Il risultato peggiore è stato ottenuto dalla coppia stereo del training set di KITTI n°105.

Figure 6.4: Immagine Left della coppia n°105

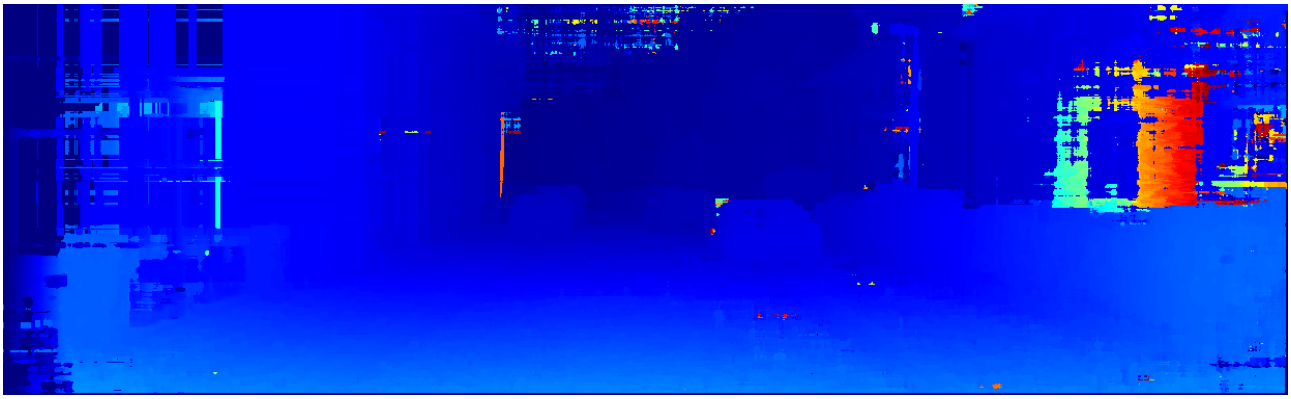


Figure 6.5: Mappa di disparità, ottenuta dall'algoritmo SGM originale

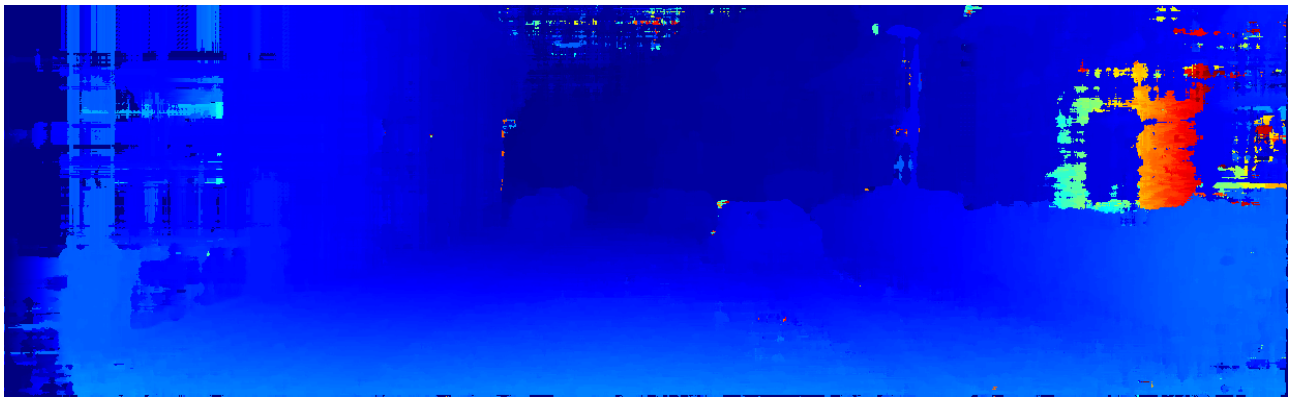


Figure 6.4: Mappa di disparità, ottenuta dal sistema modificato

In questo caso il peggioramento non è evidente come lo ero il miglioramento nel risultato di cui sopra. Tuttavia guardando i dati relativi alla coppia n°105 possiamo affermare che il sistema proposto ha apportato peggioramenti dell'ordine del 13.54% rispetto a SGM, commettendo un errore del 9.59999% contro un errore di SGM del 8.45481% [15].

6.3.2 Sistema con patch rettangolari e ortogonali e creazione di foresta di training multipla direzionale

6.3.2.1 Risultato migliore

Anche per quanto riguarda il sistema con approccio a foresta direzionale multipla il risultato migliore è stato ottenuto sulla stereo pair n°191.

Figure 6.7: Immagine Left della coppia n°191

Figure 6.8: Mappa di disparità, ottenuta da SGM

Figure 6.9: Mappa di disparità, ottenuta dal sistema modificato

In questo caso, ancor più che nel caso **6.3.1.1** le modifiche al sistema hanno apportato benefici evidenti: l'efficacia di SGM è stata incrementata del 35.81%, con errore commesso del 13% contro l'errore di SGM del 20.2642% [15].

Dal momento che in tutti e due gli scenari che stiamo esaminando i risultati migliori sono dati dalla coppia n°191, è interessante fare un confronto diretto fra le due mappe di disparità risultanti.

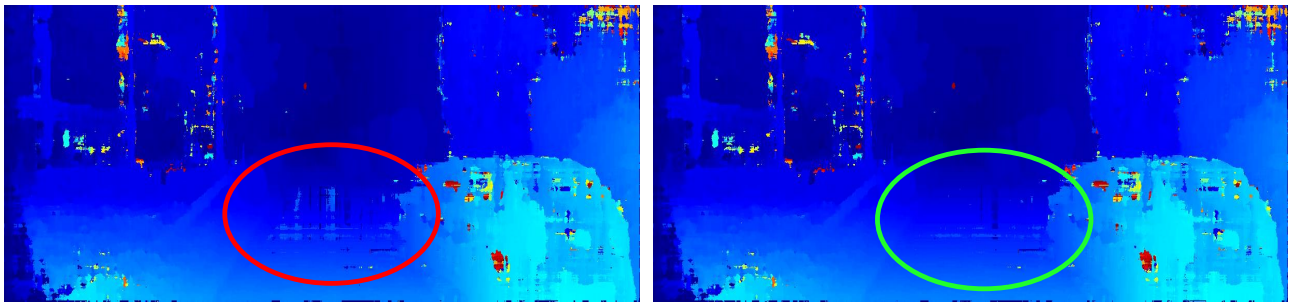


Figure 6.10: Confronto fra i risultati migliori dei due scenari in esame, rispettivamente sotto-capitolo 6.3.1.1 e 6.3.2.1

6.3.2.2 Risultato peggiore

Il risultato peggiore è ottenuto dalla coppia di immagini n°140.

Figure 6.11: Immagine Left della coppia n°140

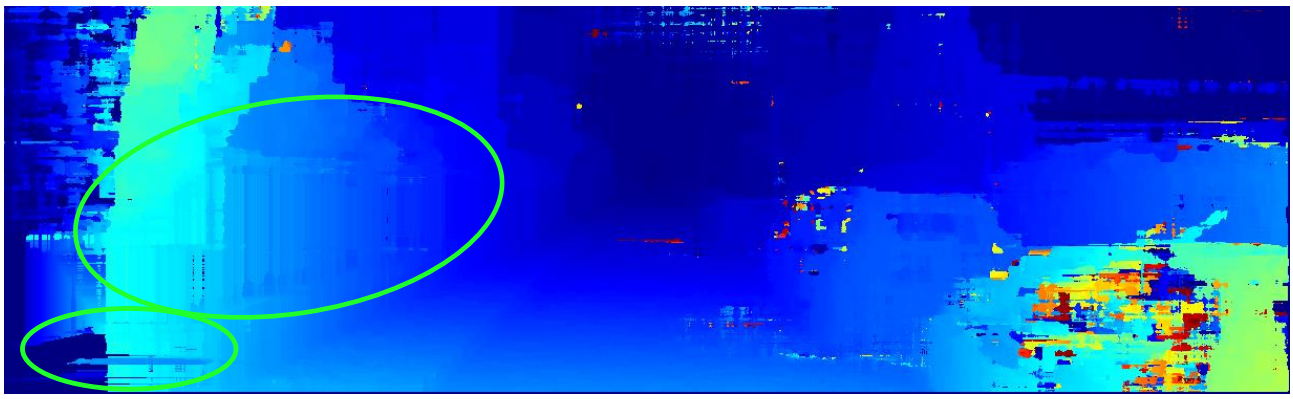


Figure 6.12: Mappa di disparità, ottenuta da SGM

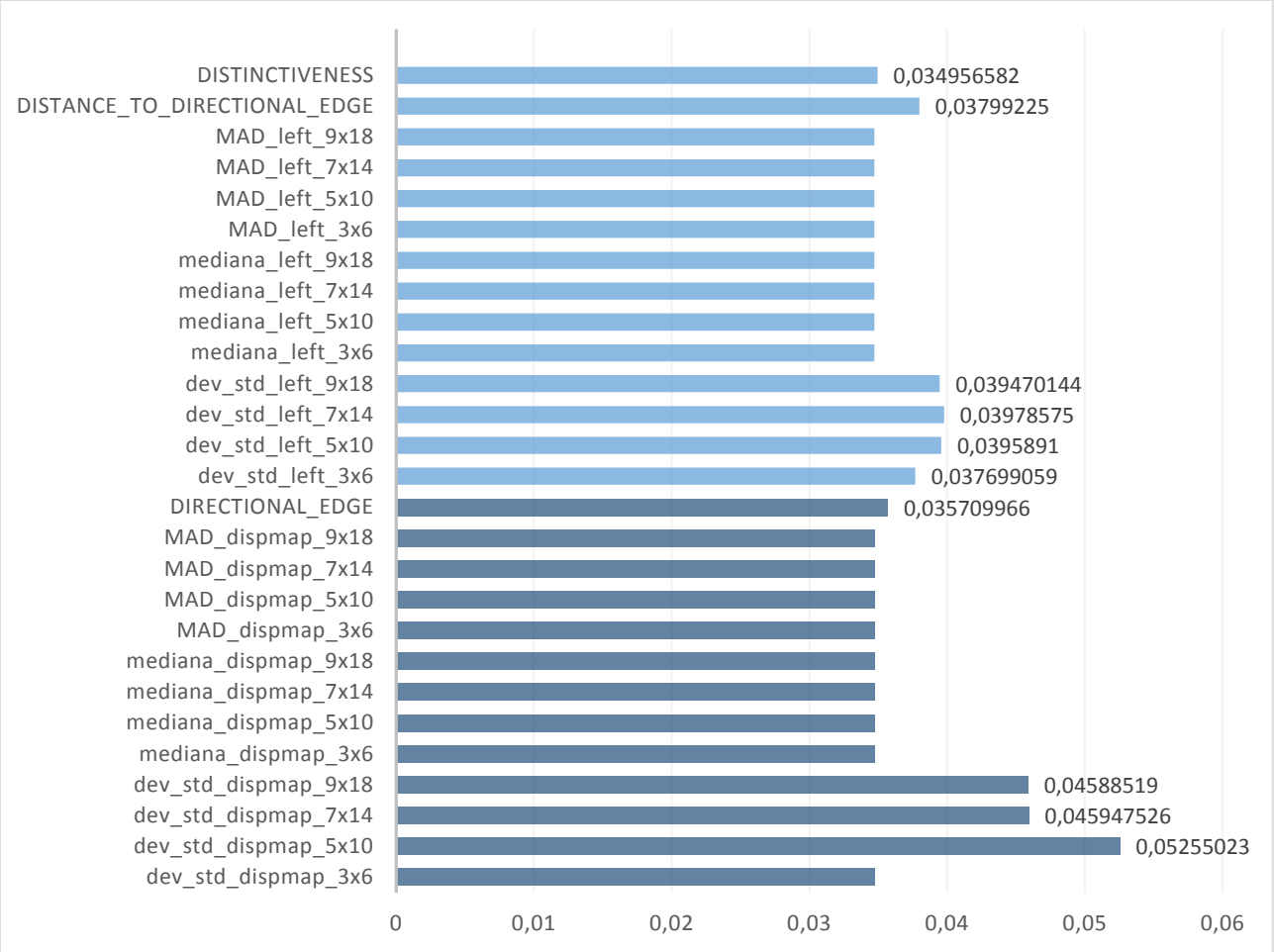
Figure 6.13: Mappa di disparità, ottenuta dal sistema modificato

In questo caso i peggioramenti introdotti dalle modifiche al sistema sono piuttosto evidenti. L'errore (cerchiato nelle Figure 6.12 e 6.13) anche qui è causato da un forte riflesso della luce, che si infrange sul muro bianco, tuttavia in questo caso le modifiche non sono state in grado di rimediare. Si ha un peggioramento ben del 98.59%, con errore commesso del 21.6647% contro un errore del 10.909% di SGM standard.

6.4 VALUTAZIONE DELL'IMPORTANZA DELLE FEATURE CALCOLATE

Le Random Forest possono essere usate anche per assegnare un punteggio agli attributi usati nel training della foresta, al fine di comprendere meglio quali siano i più significativi fra quelli del set di feature calcolate. Feature a cui viene assegnato un punteggio maggiore sono considerate più importanti rispetto quelle a cui viene assegnato un punteggio minore. Tale processo viene svolto dalla funzione ***get_var_importance()*** messa a disposizione dal componente ***CvRTrees*** di OpenCV.

Andiamo ora a valutare l'importanza delle feature utilizzate in fase di training e testing del sistema. Ricordiamo che sono in totale 27: la mediana, la deviazione standard, la deviazione assoluta mediana sia su mappe di disparità che su Left Image calcolate su finestre di 4 dimensioni diverse; la distanza di un pixel dal successivo edge direzionale e la distintività su Left Image; e infine gli edge direzionali sulle mappe di disparità.



Graph 6.10: Importanza features sistema ad 1 foresta

6.4.1 Importanza feature sistema ad 1 foresta

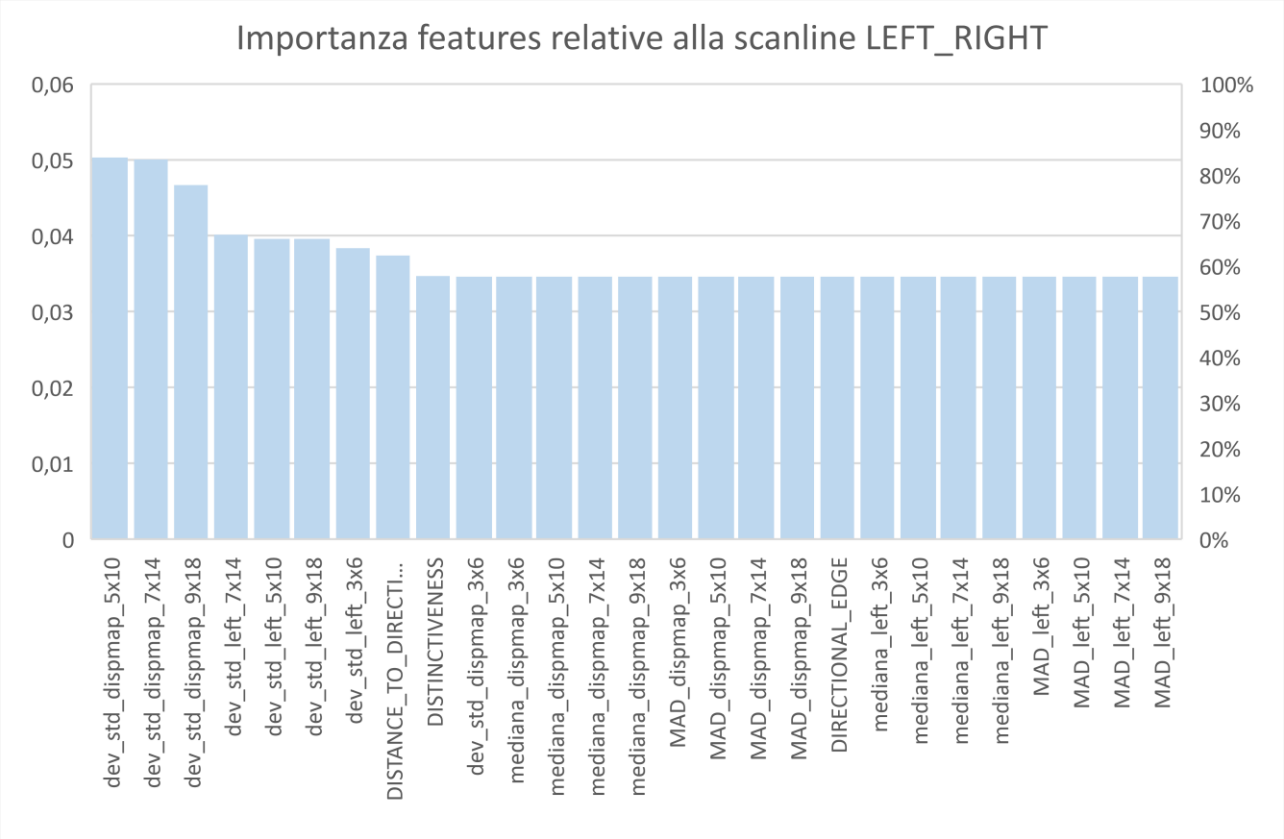
Il grafico 6.10 riporta sulle ordinate tutte le varie feature, divise in base al tipo di immagine a cui sono applicate (in azzurro quelle relative a Left Image, in blu a Mappe di Disparità) e sulle ordinate un numero compreso tra 0 e 1 che rappresenta l'importanza di tale attributo espressa in percentuale.

Possiamo quindi notare che le feature maggiormente influenti, in generale, sono rappresentate dalla deviazione standard, a prescindere dalla dimensione della patch di supporto, e in particolare le più importanti sono quelle calcolate sulle mappe di disparità. Al contrario invece le feature MAD e mediana hanno tutte la stessa

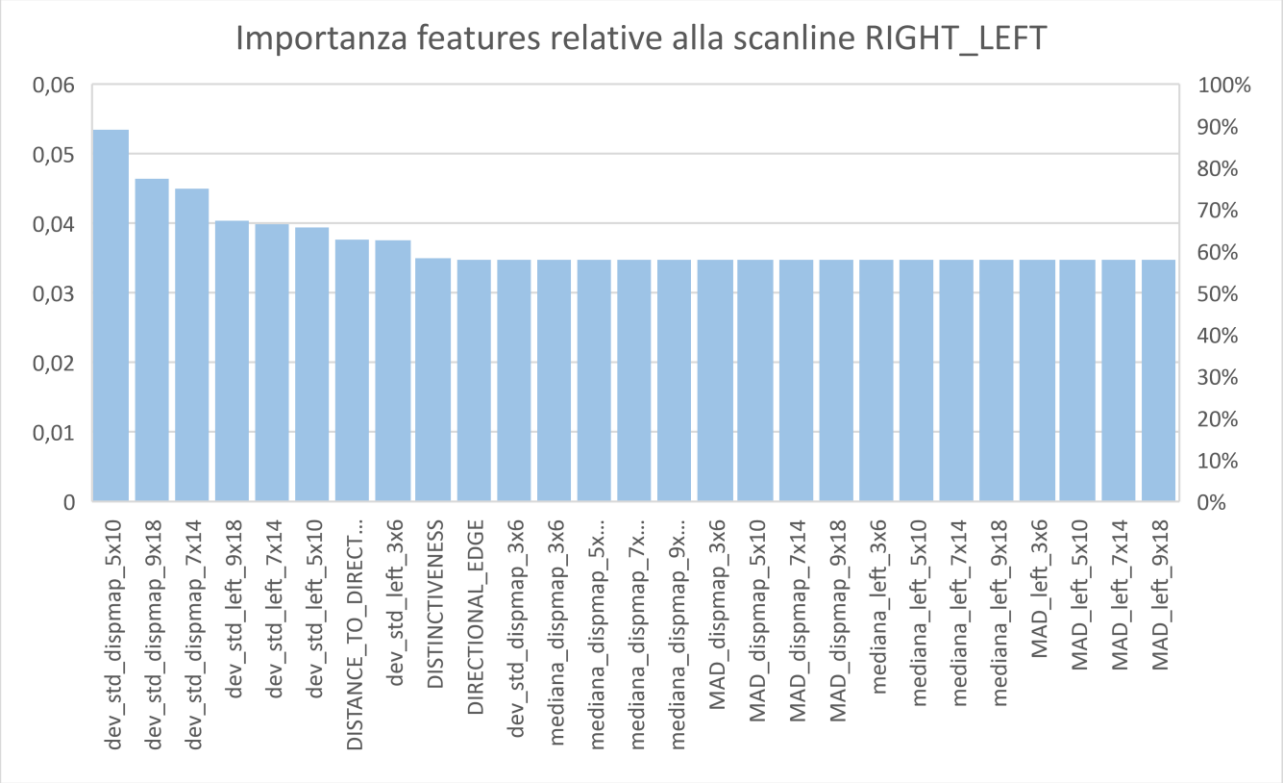
importanza (pari a 0.034730248) sia su Left Image che su mappe di disparità. Questo può far presupporre che non apportino alcun tipo di beneficio al sistema.

Focalizzandoci ora sulle nuove feature proposte per il sistema, quali *distanza dal prossimo edge direzionale*, *distintività* e *edge direzionale*, notiamo che seppure non di molto si distinguono in positivo dalle feature meno influenti, in particolare *distanza dal prossimo edge direzionale* calcolata sulle Left Image si dimostra la più importante tra le feature introdotte.

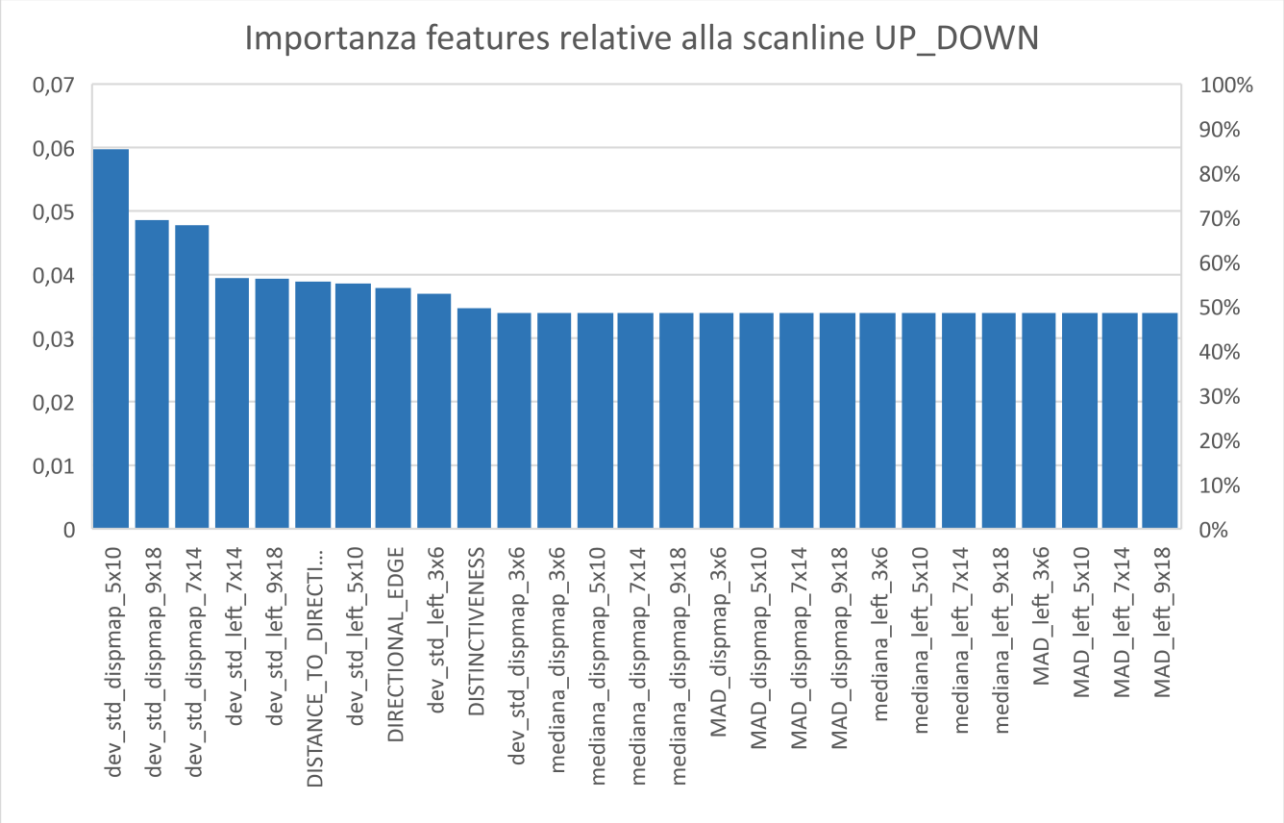
6.4.2 Importanza feature sistema a foresta multipla



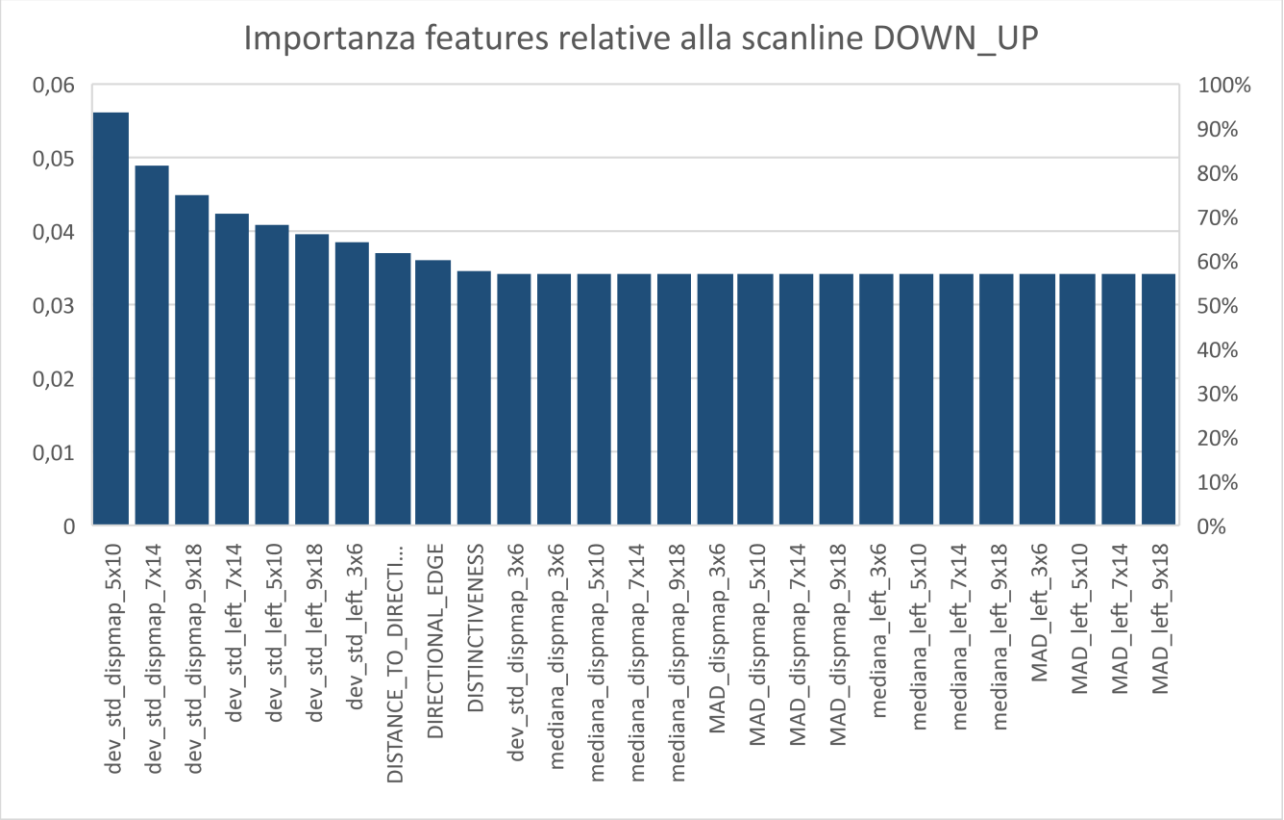
Graph 6.11: Importanza features scanline 0



Graph 6.12: Importanza features scanline 1



Graph 6.13: Importanza features scanline 2



Graph 6.14: Importanza features scanline 3

Anche in questi casi, congruentemente alle importanze delle feature ottenute dal sistema a singola foresta, le feature maggiormente influenti sono generalmente quelle rappresentate dalla deviazione standard, su mappa di disparità e Left Image, seguite dalle nuove feature, *distanza dal prossimo edge direzionale*, *distintività* e quasi sempre da *edge direzionale* (nella scanline LEFT_RIGHT infatti vediamo che *edge direzionale* è piuttosto in fondo all'asse delle ascisse, diversamente dal resto delle scanline).

Essendo ora però in gioco la direzionalità relativa alla scanline, il diverso ordine delle feature più importanti evidenzia che generare un training apposito per ogni direzione è significativo, dal momento che sono mostrati comportamenti diversi in base appunto alla scanline specifica.

CAPITOLO 7

7 CONCLUSIONI

In conclusione, l'impostazione direzionale del framework SGM_Random_Forest ha globalmente apportato miglioramenti alla già elevata accuratezza dell'algoritmo SGM tradizionale, sebbene in misura diversa a seconda dello scenario in esame.

Ad esempio, medesimi scenari (sistema a singola o multipla foresta) hanno dato esiti opposti se applicati a dataset differenti, anche se pur sempre positivi. Nel caso del dataset Middlebury l'esito migliore è stato ottenuto dal sistema a foresta multipla direzionale, mentre in KITTI è stato ottenuto da un sistema addestrato con singola foresta. Questa differenza nei comportamenti dei dataset, come già spiegato in precedenza, può essere imputabile alla natura delle scene rappresentate, estremamente diverse tra loro.

Un esempio invece di miglioramento globale, lo si può individuare nella riduzione in dimensione delle finestre di supporto, e nell'introduzione della forma rettangolare e ortogonale di queste ultime, più conforme ad un sistema che estrapola informazioni dalle immagini in modo prettamente direzionale, seguendo l'andamento delle scanline.

Un'informazione fondamentale la si ricava dallo studio delle variabili importanza fornite dalla Random Forest. Dai risultati ottenuti è evidente come alcune feature apportino benefici maggiori rispetto ad altre. Quest'informazione potrebbe risultare molto utile in vista di sviluppi futuri del presente lavoro, in quanto come già accennato precedentemente, la selezione di feature significative è cruciale ai fini della buona riuscita dell'apprendimento automatico di un sistema. Per questo si consiglia di continuare lo sviluppo di tale framework effettuando nuovi test con un sub-set delle feature utilizzate, facendo selezione fra le più importanti (evidenziate nei grafici di cui sopra); in generale esse sono la deviazione standard, la distanza dal più prossimo edge direzionale, la distintività nelle Left Image e l'individuazione degli edge direzionali nelle mappe di disparità.

Il framework utilizzato rappresenta uno strumento di Machine Learning e Computer Vision molto potente: del primo sfrutta la versatilità e popolarità degli Alberi Decisionali che compongono le RandomForest, del secondo l'accuratezza dell'algoritmo di stereo matching SGM.

La possibilità di introdurre nuove feature, unita al comportamento parametrico degli elementi del sistema, rendono il framework facilmente estendibile e riconfigurabile; gli eventuali sviluppi delle sue funzionalità sono quindi innumerevoli, alcuni esempi

dei quali possono essere considerare un numero maggiore di scanlines (16) o usare altri tipi di random forest (extremly randomized forest).

8 BIBLIOGRAFIA

[1] F. Tombari, «Algoritmi per la corrispondenza stereo».

[2] H. Hirschmueller, «Stereo Processing by Semi-Global Matching and Mutual Information,» *PAMI,* vol. 30, n. 2, pp. 328-341, 2008.

[3] C. Banz, P. Pirsch e H. Blume, «Evaluation of Penalty Functions for Semi-Global Matching Cost Aggregation,» *ISPRS,* 2012.

[4] P. Medici, «Computer Engineering Group - Università Degli Studi di Parma,» [Online]. Available: http://www.ce.unipr.it/people/medici/geometry/node123.html.

[5] S. Mattoccia, «Stereo Vision: Algorithms and Applications,» 2013.

[6] Y.-L. Chen e J.-J. Jaw, «Optimization of penalty functions for semi-global matching cost aggregation,» *Asian Association on Remote Sensing,* 2013.

[7] S. Nigro, «Metodologie di Machine Learning Applicate allaVisione Stereo,» 2014/2015.

[8] T. Mitchell, Machine Learning, McGraw Hill, 1997.

[9] S. J. Russel e P. Norvig, Artificial Intelligence: A Modern Approach.

[10] L. Breiman, «Random Forests,» *Machine Learning,* vol. 45, n. 1, pp. 5-32, 2001.

[11] «OpenCV,» [Online]. Available: http://opencv.org/.

[12] A. Spyropoulos, N. Komodakis e P. Mordohai, «Learning to Detect Ground Control Pointsfor Improving the Accuracy of Stereo Matching,» *CVPR,* 2014.

[13] [Online]. Available: http://docs.opencv.org/master/d2/d2c/tutorial_sobel_derivatives.html#gsc.tab=0.

[14] I. S. K. Kuk-Jin Yoona, «Distinctive Similarity Measure for stereo matching underpoint ambiguity,» *Computer Vision and Image Understanding,* vol. 112, n. 2, pp. 173-183, 2008.

[15] M. S. Daniele, «RIVALUTA.it,» 2002. [Online]. Available: http://www.rivaluta.it/calcola_variazione_percentuale.asp.

[16] The University of Auckland, «www.cs.auckland.ac.nz,» [Online]. Available: https://www.cs.auckland.ac.nz/courses/compsci773s1t/lectures/773-GG/figures/simplestereoB.gif.

[17] C. Stentoumis, G. Karras e E. Karkalou, «A review and evaluation of penalty functions for Semi-Global Matching,» in *IEEE International Conference on Intelligent Computer Communication and Processing*, At Cluj-Napoka, Romania, 2015.