UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in Ingegneria dell'Informazione

MISURE DI CONFIDENZA PER ALGORITMI DI STEREO VISION

Laureando

Nicola Dal Lago

Relatore

Prof. Pietro Zanuttigh

Correlatore

Giulio Marin

ANNO ACCADEMICO 2013/2014

Abstract

Lo scopo di questa tesi è il confronto di varie tecniche di stima della confidenza a partire dai dati acquisiti da una coppia di videocamere. Vengono quindi utilizzate diverse metriche, valutate sia singolarmente che combinate opportunamente. Si confrontano poi i risultati con le mappe di disparità (*ground truth*) disponibili nei dataset di riferimento. Per il calcolo delle disparità è stato usato l'algoritmo *Semi-global matching (SGM)* opportunamente modificato.

Indice

Capitolo 1

Introduzione

La visione stereo è un'area attiva della della ricerca da decenni. Negli ultimi anni, gli algoritmi di stereo vision sono maturati a tal punto da essere applicati in un vasto scenario, dall'automazione industriale, al gaming fino alla guida assistita [2].

1.1 Stereopsi

La stereopsi è la capacità percettiva che consente di unire le immagini provenienti dai due occhi, che a causa del loro diverso posizionamento strutturale, presentano uno spostamento laterale. Questa disparità viene sfruttata dal cervello per trarre informazioni sulla profondità e sulla posizione spaziale dell'oggetto mirato. Di conseguenza la stereopsi permette di generare la visione tridimensionale. *

Si possono quindi identificare i due problemi principali: calcolo delle corrispondenze e triangolazione [1].
Il primo consiste nell'accoppiare punti delle due immagini, detti punti coniugati, che sono proiezione dello stesso punto della scena nelle due viste. Il calcolo delle corrispondenze è un problema ben definito in quanto le due immagini differiscono di poco, un punto della scena deve apparire simile nei punti coniugati delle due immagini. Per semplificare il problema e ridurre il numero di errori le due immagini vengono rettificate prima del calcolo delle corrispondenze, in modo che due punti coniugati si trovino sulla stessa retta (detta retta epipolare). Questo si ottiene ruotando le immagini originali attorno ai loro centri ottici finché i piani focali non diventano co-planari (e quindi anche i piani immagine).
Per triangolazione si intende il calcolo della distanza tra un punto della scena e il piano formato dalle due fotocamere. Nel caso di due fotocamere parallele ed allineate ci si può facilmente ricondurre alla figura 1.1.
Fissato come riferimento la fotocamera di sinistra si possono scrivere le equazioni di proiezione prospettica:

*`https://it.wikipedia.org/wiki/Stereopsi`

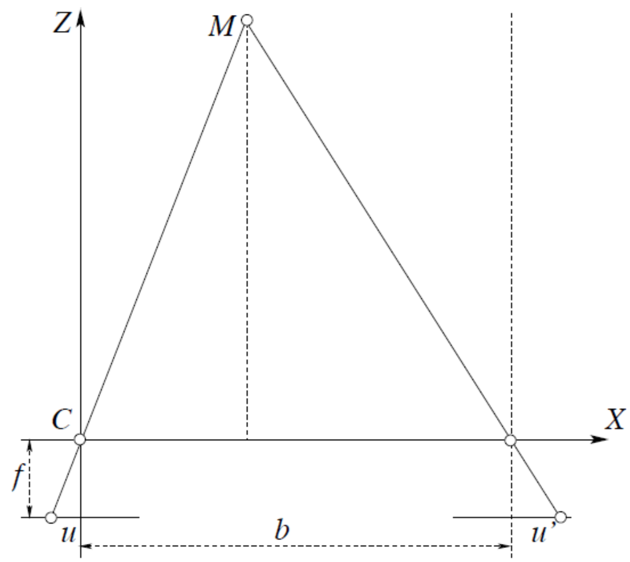


Figura 1.1: Triangolazione stereoscopica.

$$\begin{cases} \frac{f}{z} = \frac{-u}{x} \\ \frac{f}{z} = \frac{-u'}{x-b} \end{cases} \tag{1.1}$$

e risolvendo si ottiene:

$$z = \frac{bf}{u' - u} \tag{1.2}$$

dove b è la distanza tra i due centri ottici (*baseline*), f la focale delle fotocamere e $u' - u$ la disparità.

1.2 Calcolo delle corrispondenze

Il calcolo delle corrispondenze o della disparità è il problema principale della stereo vision. La disparità è la differenza tra due punti coniugati (vettore), immaginando di sovrapporre le due immagini. Il calcolo delle corrispondenze non è altro che il calcolo della disparità per ogni pixel delle due immagini [1]. Si ottiene quindi una mappa di disparità del tipo di figura 1.2c. La mappa di disparità non è altro che una matrice di scalari: in ogni cella viene memorizzato la distanza che separa il corrispondente punto dell'immagine di riferimento dal suo punto coniugato.
Esistono molti algoritmi diversi per il calcolo delle disparità. I metodi *locali* utilizzano per la ricerca del punto coniugato, un piccolo numero di pixel attorno al pixel

(a) *Fotocamera di destra.* (b) *Fotocamera di sinistra.* (c) *Mappa di disparità.*

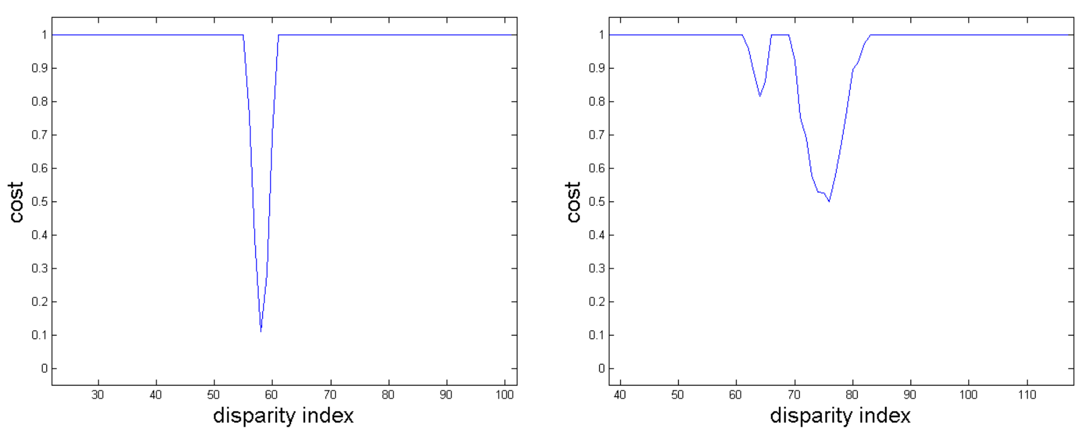
Figura 1.2: Mappa di disparità con immagine di destra come riferimento, immagine presa da `http://vision.middlebury.edu/stereo/data/`.

considerato, approcci tipici di questo tipo sono *correlazione*, *SSD*, *SAD*, *trasformata census*, eccetera [3]. I metodi *globali*, invece, sfruttano vincoli non locali per ridurre la sensibilità a regioni per le quali l'accoppiamento fallisce. Ne risulta un problema di ottimizzazione e richiedono un costo computazionale sicuramente maggiore rispetto ai metodi locali. In questa tesi si utilizzano invece dei metodi detti *semi globali*, che non sono altro che una via di mezzo tra i locali e globali; questi considerano un pixel e i suoi vicini per approssimare la soluzione globale. Vengono utilizzati questi tipi di algoritmi sia per ridurre il costo computazionale rispetto ai globali, sia per avere migliore precisione rispetto ai locali [4]. La disparità però non è una funzione esatta, in quanto può essere soggetta da rumore (errata illuminazione, mancanza di trama, occlusioni, eccetera) provocando errori nell'algoritmo. Viene quindi generata una funzione detta " funzione costo ", la quale ha per ascissa tutti i possibili valori di disparità, e per ordinata il valore di costo, che indica quanto quella disparità sia attendibile.

Capitolo 2

Misure di confidenza

Come accennato nella sezione 1.2, ad ogni pixel dell'immagine di riferimento (destra o sinistra), viene assegnata una funzione costo, la quale identifica quanto una determinata disparità sia esatta per quel pixel.



(a) *Funzione costo ideale.* (b) *Funzione costo ambigua.*

Figura 2.1: Due possibili funzioni costo.

Due possibili funzioni costo sono riportate in figura 2.1. E' chiaro che nel primo caso la disparità è individuata da quell'unico picco; nel secondo caso invece, la funzione è ambigua, in quanto vi sono diversi picchi, e non è quindi più così immediato trovare l'esatta disparità. Si rende quindi necessario lo studio di varie tecniche per misurare la confidenza con la quale viene assegnato un determinato valore. Per il calcolo della funzione costo di ciascun pixel, è stato usato un algoritmo di tipo *semi-global matching*, presente nelle librerie open source di visione computazionale *OpenCV* [9]. Questo algoritmo, scritto nel linguaggio *C++*, di per se produrrebbe in output solo la mappa di disparità. L'algoritmo è stato quindi opportunamente modificato per poter estrarre la funzione costo di ciascun punto. Si ha quindi in output anche una matrice, delle stesse dimensioni dell'immagine originale, ma che ha per ciascuna cel-

la la funzione costo. Successivamente la matrice viene convertita in formato *.mat*, per rendere il tutto più facile da utilizzare con il linguaggio di programmazione *MATLAB*. Poi, con script *MATLAB*, vengono calcolate le confidenze con diverse metriche. Le immagini utilizzate sono state prese dal dataset *Midleburry* [7, 8] disponibili su `http://vision.middlebury.edu/stereo/data/`.
Prima di proseguire però, si rende necessario dare alcune definizioni:

- $c(d)$: il valore del costo assegnato ad una ipotetica mappa di disparità d, per un pixel di coordinate (x, y), è denotato con $c(x, y, d)$ o $c(d)$, se le coordinate del pixel non sono ambigue;
- c_1: il minimo valore del costo di un pixel è definito con c_1 e il corrispondente valore di disparità da d_1; $c_1 = c(d_1) = \min\{c(d)\}$;
- c_2: con c_2 viene denotato il secondo valore più piccolo in d_2, mentre con c_{2m} si denota il secondo minimo locale più piccolo. Nel calcolo del secondo minino, bisogna anche tener conto di quei casi dove vi sono due minimi molto vicini e con valori molto simili; in questo caso si escludono minimi che non sono ad una certa distanza dal minimo principale.

Seguono quindi tutte le varie tecniche utilizzate, raggruppate secondo l'aspetto del costo che considerano [6].

2.1 Proprietà locali della curva

In questo genere di metriche, si sfrutta il fatto che la forma della curva di costo intorno al minimo (la nitidezza o la planarità) è indice di certezza nel confronto.

Curvature (CUR) E' largamente utilizzata nella letteratura [6], ed è definita come

$$C_{CUR} = \frac{-2c(d_1) + c(d_1 - 1) + c(d_1 + 1)}{2} \tag{2.1}$$

se $d_1 - 1$ o $d_1 + 1$ sono fuori dal range di disparità, il punto minimo $c(d_1)$ viene usato due volte. Il tutto viene diviso per 2 per garantire valori compresi tra zero e uno.

Local Curve (LC) Molto simile alla misura *CUR*, la *Local Curve* è descritta da

$$C_{LC} = \frac{\max\{c(d_1 - 1), c(d_1 + 1)\} - c_1}{\gamma} \tag{2.2}$$

il parametro γ verrà poi scelto tale da garantire una distribuzione del costo più uniforme possibile tra zero e uno.

2.2 Minimo locale della curva

Si basa sul concetto che la presenza di altri candidati è un'indicazione di incertezza, mentre la loro assenza di certezza.

Peak Ratio Naive (PKRN) A differenza del *Peak Ratio (PKR)*, che calcola il costo con la formula 2.3

$$C_{PKR} = \frac{c_{2m}}{c_1} \tag{2.3}$$

il *PKRN* non richiede che il numeratore sia un minimo locale. Inoltre la formula è leggermente diversa da quella proposta in letteratura [2]

$$C_{PKRN} = \frac{c_2 + \epsilon}{c_1 + \epsilon} - 1 \tag{2.4}$$

PKRN può essere visto come una combinazione del *PKR* e *CUR*, che assegna bassa confidenza per le corrispondenze con minimi piatti o concorrenti forti. Anche se le modifiche al *PKRN* violano leggermente la metrica originale, ha i seguenti vantaggi rispetto alla controparte originale:

- le rare singolarità in cui il denominatore sia nullo non sono più presenti;
- piccole variazioni nei costi dovuti al rumore ai livelli bassi del costo, non hanno un forte impatto nella metrica;
- potendo scegliere il valore di ϵ, il range dei possibili valori può essere adattato, al fine di omogeneizzare la misura tra zero e uno.

Maximum Margin (MMN) il margine tra c_1 e c_2 è anch'esso indicazione di confidenza. *MMN* è definito dalla formula

$$C_{MMN} = c_2 - c_1 \tag{2.5}$$

Nonlinear Margin (NLM) è definito come

$$C_{NLM} = e^{\frac{c_2 - c_1}{2\sigma^2_{NLM}}} - 1 \tag{2.6}$$

le variazioni del parametro σ_{NLM} verranno poi discusse nella sezione 3.1.

2.3 Intera curva

Questi metodi convertono la funzione costo in una distribuzione di probabilità sulla disparità.

Maximum Likelihood Metric (MLM) Insieme a *PKRN* è una delle metriche più promettenti. Entrambe hanno ottenuto risultati sopra la media sia su immagini indoor che outdoor [6]. La *Maximum Likelihood Metric* è definita come

$$C_{MLM} = \frac{e^{-\frac{c_1}{2\sigma^2_{MLM}}}}{\sum_d e^{-\frac{c(d)}{2\sigma^2_{MLM}}}} \tag{2.7}$$

In questo caso σ_{MLM} rappresenta l'incertezza della disparità, e anche per questo caso la sua variazione verrà discussa nella sezione 3.1.

Attainable Maximun Likelihood (AML) è una variante di *MLM* che modella il costo per un particolare pixel usando una distribuzione Gaussiana centrata nel minimo valore di costo che è stato finora calcolato per quel pixel (c_1 nella nostra notazione).

$$C_{AML} = \frac{e^{-\frac{(c_1-c_1)^2}{2\sigma^2_{AML}}}}{\sum_d e^{-\frac{(c(d)-c_1)^2}{2\sigma^2_{AML}}}} = \frac{1}{\sum_d e^{-\frac{(c(d)-c_1)^2}{2\sigma^2_{AML}}}} \tag{2.8}$$

Winner Margin Naive (WMNN) esiste il *Winner Margin*, che richiede che al numeratore vi sia un minimo locale, ma come per il *PKRN* definiamo la versione naive come

$$C_{WMNN} = \frac{c_2 - c_1}{\sum_d c(d)} \tag{2.9}$$

2.4 Consistenza fra disparità di destra e sinistra

Questa tipologia di misure consiste nel fatto che, idealmente, un punto nella mappa di disparità destra dovrebbe essere lo stesso nella mappa di disparità sinistra. Fin ora abbiamo denotato il valore del costo con $c(x, y, d)$ avendo come riferimento l'immagine di sinistra, ma per chiarezza chiameremo $c_R(x_R, y, d_R)$ il valore del costo ottenuto tenendo come riferimento l'immagine di destra. Inoltre se d è il valore di disparità dell'immagine di sinistra, d_R è quello dell'immagine di destra. Questo genere di misure risulterà più complicata delle precedenti, in quanto prima di calcolare la funzione costo, si rende necessario calcolare prima la disparità sia per l'immagine di destra che quella di sinistra.

Left Right Consistency (LRC) *Left Right Consistency (LRC)* viene calcolato prendendo il valore disparità calcolato in una immagine, e proiettandolo nell'altra immagine. Se la differenza nei valori è inferiore a una determinata soglia, allora il pixel è occluso. Questo procedimento viene poi ripetuto anche al contrario, cioè proiettando la seconda immagine nella prima. Inoltre il costo di *LRC* viene calcolato come segue:

$$C_{LRC}(x,y) = \big|d_1 - D_R(x-d_1,y)\big| \tag{2.10}$$

con $d_1 = \arg\min_d\big\{c(x,y,d)\big\}$ e $D_R(x-d_1,y) = \arg\min_{d_R}\big\{c_R(x-d_1,y,d_R)\big\}$.

Left Right Difference (LRD) *Left Right Difference (LRD)* è una misura proposta per la prima volta in [5]. Essa considera sia i due minimi della disparità di sinistra, ma anche il minimo di quella di destra. E' definita come

$$C_{LRD}(x,y) = \frac{c_2 - c_1}{\big|c_1 - \min\{c_R(x-d_1,y,d_R)\}\big|} \tag{2.11}$$

L'idea è che finestre di pixel corrispondenti dovrebbero risultare in valore di costo molto simili, e quindi piccoli valori al denominatore. Questa misura dovrebbe salvaguardare da due possibili errori:

- se il margine $c_2 - c_1$ è grande, ma il pixel ha una corrispondenza errata, il denominatore sarà grande, e la confidenza bassa;
- se il margine è piccolo, la misura è possibile che sia ambigua, in questo caso se il denominatore è piccolo denota che è stata creata con successo una corrispondenza fra due pixel

Capitolo 3

Risultati

Tutte le misure effettuate vengono confrontate con una mappa di disparità *ground truth* che contiene i veri valori di disparità per ogni pixel. Questa mappa è sempre disponibile nei dataset utilizzati [7, 8], per tutte le immagini e per tutte le risoluzioni. La disparità *ground truth* viene calcolata utilizzando sempre una coppia di videocamere, ma aggiungendo anche uno o più proiettori che illuminano la scena con dei pattern specifici. Ogni telecamera utilizza quindi i pattern per determinare un codice univoco per ciascun pixel. Trovare quindi le corrispondenze si traduce banalmente nel verificare quali pixel delle due immagini hanno lo stesso codice [10].

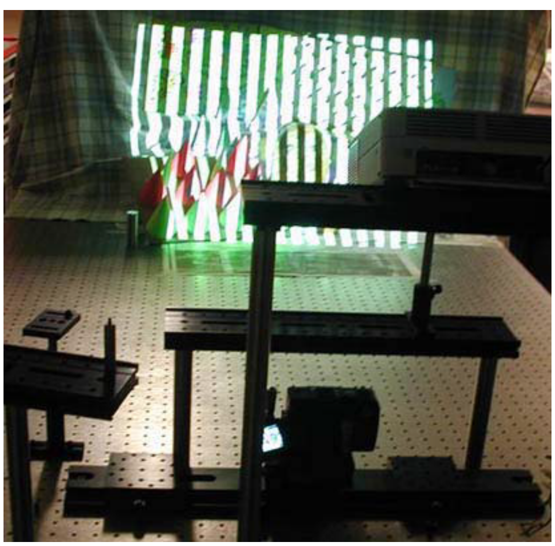


Figura 3.1: Configurazione del proiettore e della coppia di videocamere per il calcolo della disparità ground truth; si può notare come il proiettore illumina la scena con dei pattern diversi per facilitare il compito delle videocamere.

Prima di continuare definiamo l'errore di disparità come

$$e_d = |d_{GT} - d| \tag{3.1}$$

Dove d_{GT} è la disparità *ground truth* e d sono i valori di disparità ottenuti con l'algoritmo *SGM*.

Per valutare la capacità delle misure di confidenza di predire dove una disparità è corretta, è stato creato uno script *MATLAB* che ordina prima l'errore di disparità in ordine crescente, e poi ordina l'errore per ogni misura in ordine decrescente rispetto la sua confidenza. Successivamente si calcola l'errore globale selezionando prima il 5%, in due modi:

1. nel primo metodo si misura la percentuale di pixel che ha un errore di disparità maggiore di uno, questo si fa perché ovviamente se l'errore è minore di uno, non vi è alcun pixel di differenza fra le due;

2. nel secondo si misura semplicemente l'errore medio.

Si ripete poi con il 10% e così via sino al 100%. Prima di confrontare le varie metriche si è pero' trovato il parametro ottimale per quelle con parametri variabili.

3.1 Variazione parametri

In questa sezione, prima di confrontare le misure, verificheremo quali parametri sono più adatti per quelle che dipendono da tali. Consideriamo quindi la *Local Curve*, *Peak Ratio Naive*, *Nonlinear Margin*, *Maximum Likelihood metric* e *Attainable Maximun Likelihood*.

Variazione parametro γ per *LC* Ora indagheremo quali effetti produce la variazione del parametro γ. Come prima cosa si è osservato per quale valore si ha una distribuzione più uniforme tra 0 e 1. Come si può vedere dalla figura 3.2, si ha una

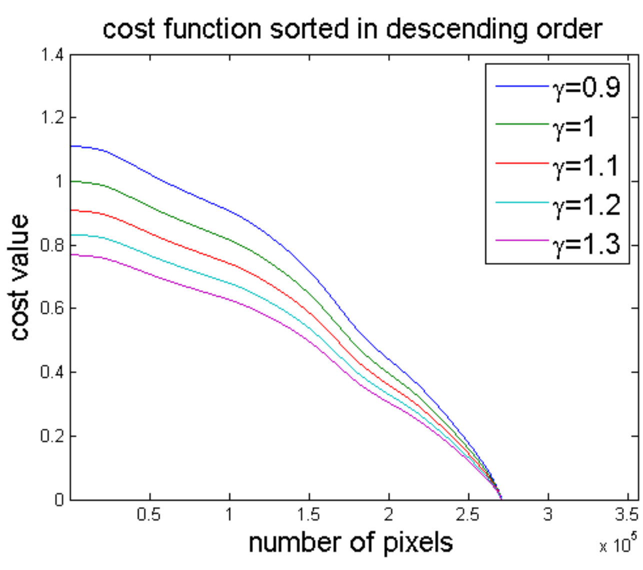


Figura 3.2: Distribuzioni diverse con valori di γ diversi.

migliore distribuzione per $\gamma = 1$. Per quanto riguarda l'errore medio invece, non si ottiene nessun cambiamento significativo sia con valori di γ molto alti, sia con valori molto bassi. Come valore finale si è quindi deciso per un γ pari a 1; facendo ciò non sarà anche necessario riscalare la funzione costo, in quanto è già compresa tra zero e uno.

Variazione parametro ϵ per *PKRN* Come prima, verifichiamo come si comporta la distribuzione della funzione al cambiare del parametro ϵ. Si nota che per valori intorno a uno, la distribuzione non supera il massimo di uno, con valori minori di uno invece lo si supera. Per quanto riguarda gli errori rispetto la *ground truth*, si notano dei lievi miglioramenti per valori di ϵ molto piccoli, intorno a un decimo; lo si può notare in figura 3.3. Si è notato inoltre che dopo il valore di $0,128$ non si hanno cambiamenti. Si sceglie quindi $\epsilon = 0.128$. Notare che in questo modo è necessario riscalare il valore della metrica, in quanto il suo massimo è abbondantemente sopra l'uno; per farlo è sufficiente dividere ogni singolo costo per il costo massimo.

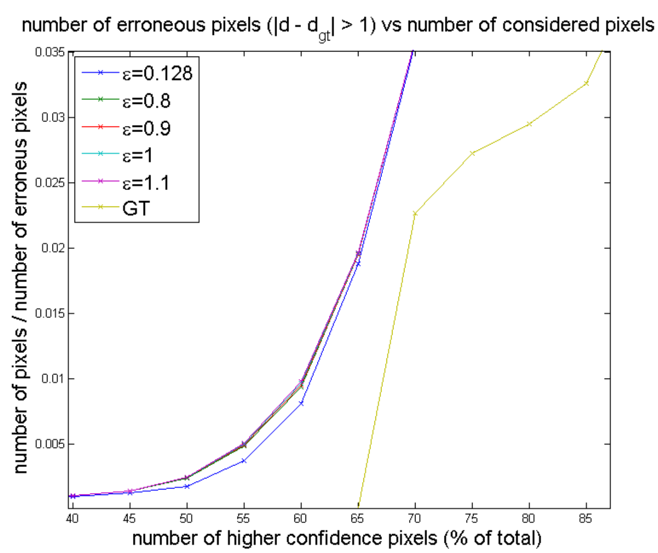


Figura 3.3: Dettaglio della curva dell'errore della metrica *PKRN*, notare che con ϵ più piccolo si hanno dei lievi ma visibili miglioramenti; in giallo il valore di *ground truth*

Variazione parametro σ per *NLM* Ripetiamo ancora una volta il procedimento, i risultati sono pressoché simili al *LC*, cioè grosse variazioni di distribuzione e trascurabili variazioni di errore per differenti valori di σ. Dopo qualche tentativo con diversi parametri, si arriva alla conclusione che i risultati migliori si ottengono con $\sigma = 0,85$. Si fa notare però, che con valori di σ molto piccoli, l'errore può peggiorare di molto, mentre con valori molto grandi l'errore rimane pressoché inalterato.

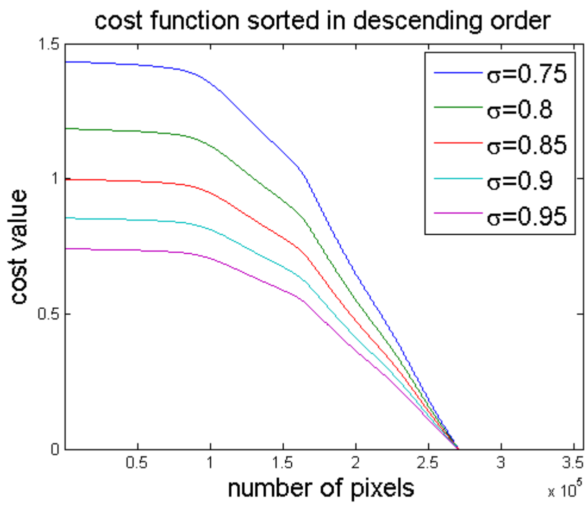


Figura 3.4: Diversi valori di distribuzione per diversi valori di σ.

Variazione parametro $\boldsymbol{\sigma}$ per *MLM* A differenze delle altre misure, per essere utilizzata al meglio la si deve per forza riscalare. Infatti i valori che permetterebbero un costo compreso tra zero e uno, sarebbero completamente sballati per quanto riguarda gli errori. Per garantire quinti una certa uniformità e un errore non esagerato, si è quindi scelto un valore di σ pari a $0,3$.

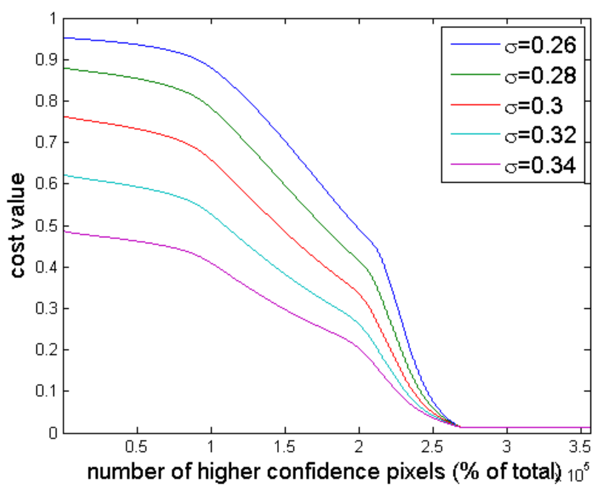
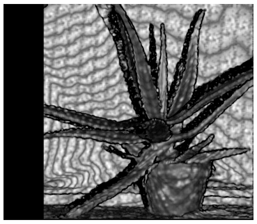


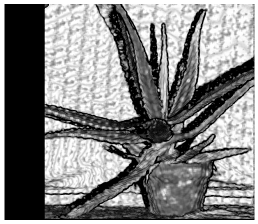
Figura 3.5: Diversi valori di distribuzione per diversi valori di σ.

Variazione parametro $\boldsymbol{\sigma}$ per *AML* Il suo comportamento è molto simile a *MLM*, i migiori risultati si ottengono con σ del valore di $0,4$.

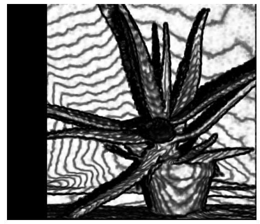
3.2 Confronto delle misure



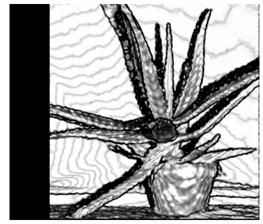
(a) *Misura Curvature.*



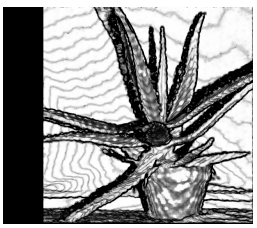
(b) *Misura Local Curve.*



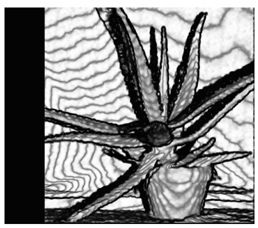
(c) *Misura Peak Ratio Naive.*



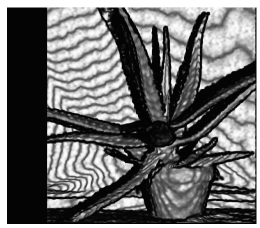
(d) *Misura Maximum Margin.*



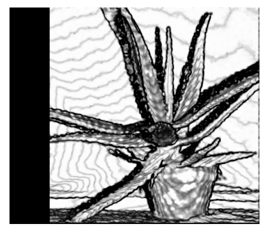
(e) *Misura Nonlinear Margin.*



(f) *Misura Maximum Likelihood metric.*



(g) *Misura Attainable Maximum Likelihood.*



(h) *Misura Winner Margin Naive.*

Figura 3.6: Diverse misure di confidenza.

Cominciamo dal dire cosa aspettarci dai due confronti descritti prima. Per il primo grafico ci aspettiamo di trovare una curva che resta a zero per una determinata percentuale di confidenze, e che poi tende a crescere in maniera pressoché lineare per le successive.

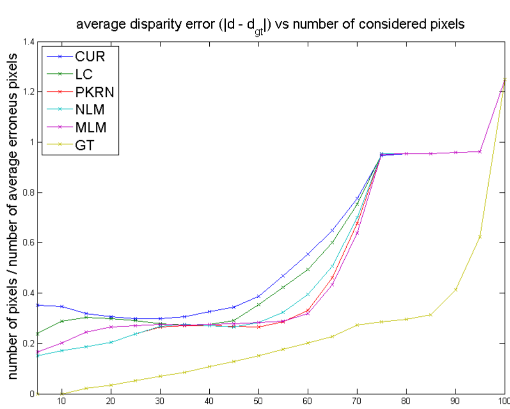
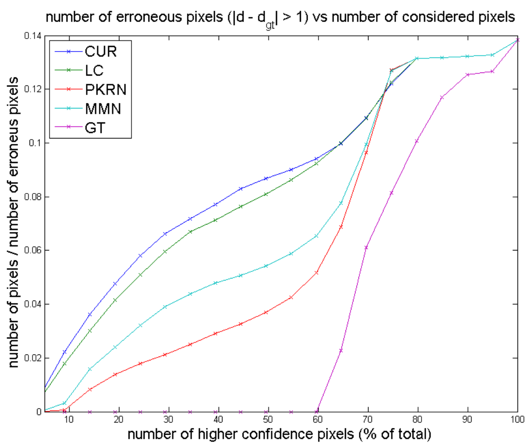
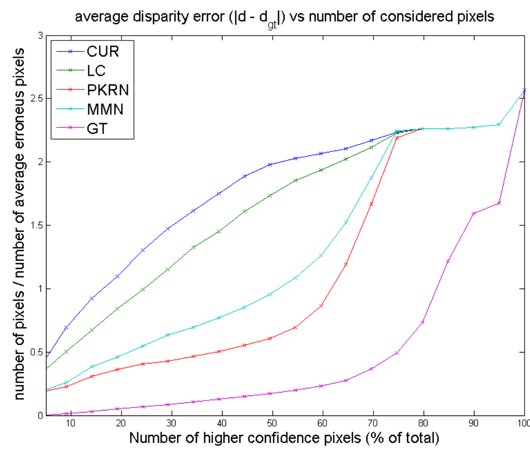


Figura 3.7: Errore di disparità medio nella figura *Aloe*.

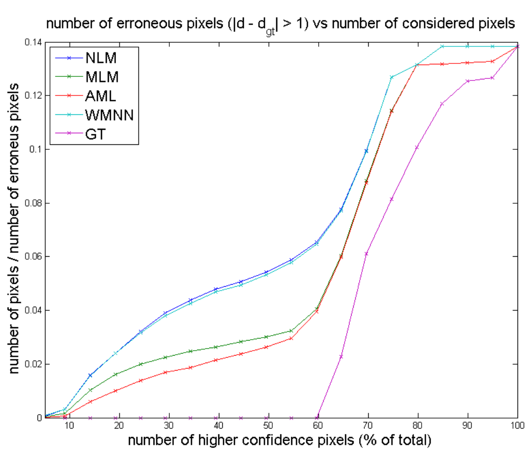
Per il secondo confronto ci aspetteremo che l'errore di riferimento sia monotono crescente, mentre le misure possono assumere andamenti più casuali, al patto che abbiano errore sempre maggiore di quello di riferimento. Ovviamente la misura migliore sarà quella con la curva il più possibile vicino all'errore di *ground truth*. Passando quindi al confronto, si nota che le diverse metriche si comportano in maniera differente a seconda dell'immagine. Nel complesso però, le due che sembrano avere risultati migliori sono la *MLM* e *AML*. Questo è chiaramente visibile nei grafici riportati in figura 3.8.Questo è il risultato dove differenzia di più tutte le misure; questo vuol dire che in certe immagini le misure si equivalgono molto, mentre in altre sono molto diverse. Non succede mai però che *MLM* e *AML* siano nettamente peggiori, succede invece che a volte una delle due sia meglio di un altra e viceversa. Altro caso, è quando i grafici si incrociano, questo vuol dire che per una certa percentuale di confidenze è meglio una, mentre poi è meglio un'altra. Questo succede ad esempio nella figura 3.7, dove la metrica *NonLinear Margin* è la migliore di tutte per quasi il 40% dei pixel considerati, venendo poi superata, ancora una volta, da *MLM* e *PKRN*. Come previsto, anche la metrica *PKRN* ottiene buoni risultati, superando tutte soprattutto nella prima percentuale di pixel considerati.



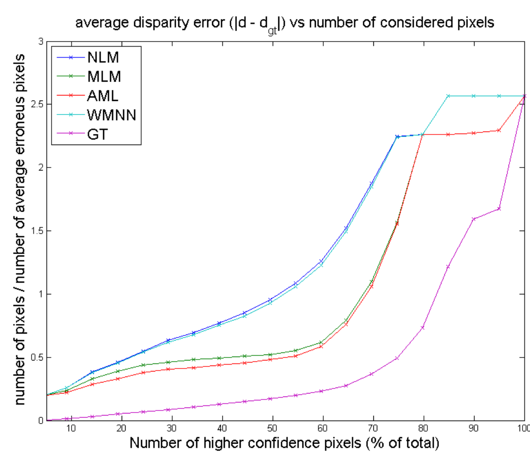
(a) *Numero di pixel con errore di disparità maggiore di uno.*



(b) *Errore di disparità medio.*



(c) *Numero di pixel con errore di disparità maggiore di uno.*



(d) *Errore di disparità medio.*

Figura 3.8: Grafici dei risultati utilizzando l'immagine *Bowling2* dal dataset *Middlebury*

Per quanto riguarda le misure del tipo consistenza fra disparità di destra e sinistra, la migliore delle due sembra essere *LRD*. Come verificato in [5], questa misura è migliore di *LRC* in praticamente tutte le immagini considerate e per quasi tutte le percentuali di pixel. Entrambe comunque si comportano molto bene anche rispetto le misure descritte nel capitolo 2; infatti la curva che indica il numero di pixel con errore di disparità maggiore di uno, rimane piatta fino a circa il 50% dei pixel considerati (il *ground truth* arriva al 65%) mentre anche nel migliore dei casi, le altre misure non riescono a superare il 30%.

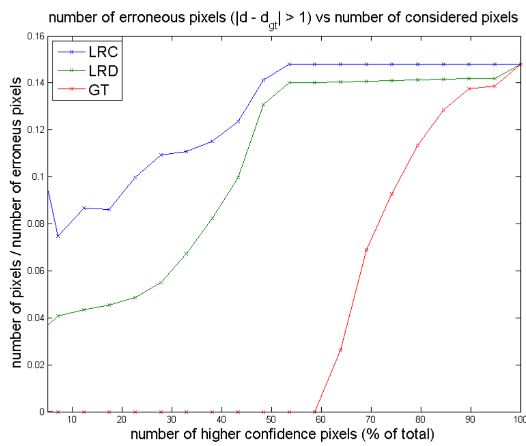


Figura 3.9: Misura *LRD* contro *LRC*.

Capitolo 4

Combinazione di misure

Per combinazione di misure, si intende il combinarne più insieme al fine di migliorarne i risultati. Considerando le misure di confidenza come probabilità, la cosa più semplice da fare sarebbe quella di dire che, essendo indipendenti, la loro combinazione non è altro che il loro prodotto. In realtà però, non è del tutto corretto definirle indipendenti, in quanto tutte dipendono dalla stessa "matrice dei costi", calcolata con il medesimo algoritmo (in questo caso *SGM*). In questa tesi però non ne terremo conto, calcolando la combinazione di misure solo come semplice moltiplicazione di alcune di esse. Come combinazione sono state usate:

- combinazione di *AML* e *MLM* $C_{comb1} = C_{AML} \cdot C_{MLM}$;
- combinazione delle tre migliori $C_{comb2} = C_{AML} \cdot C_{MLM} \cdot C_{PKRN}$;
- combinazione fra *NLM* e *MLM*, infatti la metrica *NLM* è risultata competitiva in certi dataset per circa il primo 40% dei pixel, per poi essere superata da *MLM* $C_{comb3} = C_{NLM} \cdot C_{MLM}$.

Ovviamente prima di moltiplicarle, bisogna verificare che la loro distribuzione sia compresa tra zero e uno.

4.1 Risultati

La migliore combinazione risulta essere la prima, poi la seconda e infine la terza. La prima combinazione riesce sempre ad essere migliore di *MLM*, mentre solo a volte migliore di *AML*. La seconda migliora, e non di poco, la metrica *PKRN*, mentre la terza è peggiore delle singole misure per i primi 50% di pixel, ma in quelli successivi risulta vincente. Nel complesso quindi, questo genere di combinazioni, può riuscire a migliorare i risultati di qualcosa, ma un uso sbagliato può riuscire a peggiorarli di molto.

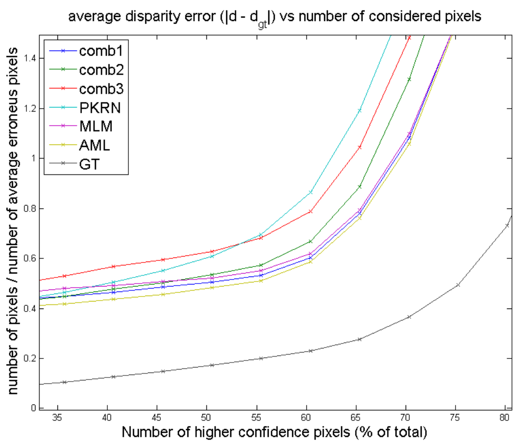


Figura 4.1: Dettaglio dell'errore medio per la combinazione di diverse misure, in questo caso la migliore resta comunque *AML*, per poi essere raggiunta dalla sua combinazione con *MLM*.

Per riuscire a migliore questi risultati, si potrebbe pensare a combinazioni più complicate, come utilizzare un approccio di tipo *machine learning*, utilizzando *random tree ensembles* descritto in [11].

Appendice A

Codice MATLAB utilizzato

In questa appendice viene riportato solo il codice *MATLAB* che calcola le mappe di confidenza con le varie tecniche descritte nel paragrafo 2 e nel paragrafo 4; si riporta inoltre il codice utilizzato per analizzare i risultati ottenuti.

A.1 compute_confidence.m

Questo script computa la mappa di disparità per *CUR*, *LC*, *PKRN*, *MMN*, *NLM*, *MLM*, *AML* e *WMNN*. Calcola inoltre la combinazione delle confidenze.

```
% This script compute confidence with different metrics

% @author: Giulio Marin (giulio.marin@me.com), Nicola Dal Lago
% @date: 25/08/2014
% @version: 1.0


%% clear workspace
clear; close all; clc;

dataset = 'Bowling2';      %it must be 'Aloe', 'Bowling2', 'Flowerpots'

rows = 555;
if(strcmp(dataset, 'Aloe'))
    cols = 641;
elseif(strcmp(dataset, 'Bowling2'))
    cols = 665;
elseif(strcmp(dataset, 'Flowerpots'))
    cols = 656;
end

load(['../../../C/data/Images/Middlebury/' dataset '/SGM/cost.mat'])

% Smallest costs and indexes
[c1,I1] = min(C,[],3);

% Second smallest costs and indexes
c2 = 32767*ones(size(c1));
I2 = ones(size(c1));
for d = 1:diff(disparity)
    index = (squeeze(C(:,:,d)) >= c1);
    index = index & (squeeze(C(:,:,d)) < c2);
```

```
    index = index & abs(I1 - d) > 1;
    if any(index(:))
        tmp = squeeze(C(:,:,d));
        c2(index) = tmp(index);
        I2(index) = d;
    end
end


%% Curvature (CUR) and Local Curve (LC)
Im = I1 - 1; Im(Im == 0) = 1;
Ip = I1 + 1; Ip(Ip == 81) = 80;

Cm = zeros(size(c1));
Cp = zeros(size(c1));

for r=1 : rows
    for c=1 : cols
        Cm(r,c) = C(r,c,Im(r,c));
        Cp(r,c) = C(r,c,Ip(r,c));
    end
end

% Curvature (CUR)
C_CUR = (-2*c1 + Cm + Cp) / 2;
figure;imshow(C_CUR);title('Curvature')

% Local Curve (LC)
gamma = 1;

C_LC = (max(Cm,Cp)-c1)./gamma;
figure;imshow(C_LC);title('Local Curve')


%% Peak Ratio Naive (PKRN)
epsilon = 0.128;

C_PKRN = (c2+epsilon)./(c1+epsilon) - 1;
C_PKRN = C_PKRN ./ max(C_PKRN(:));
figure;imshow(C_PKRN);title('Peak Ratio Naive')


%% Maximum Margin (MMN)
C_MMN = c2 - c1;
figure;imshow(C_MMN);title('Maximum Margin')


%% Nonlinear Margin (NLM)
sigma_NLM = 0.85;

C_NLM = exp((c2 - c1)./(2*sigma_NLM^2)) - 1;
figure;imshow(C_NLM);title('Nonlinear Margin')


%% Maximum Likelihood metric (MLM)
sigma_MLM = 0.3;

sum = 0;
for d = 1:diff(disparity)
    sum = sum + exp(-squeeze(C(:,:,d))./(2*sigma_MLM^2));
end

C_MLM =exp(-c1./(2*sigma_MLM^2))./sum;
C_MLM = C_MLM ./ max(C_MLM(:));
figure;imshow(C_MLM);title('Maximum Likelihood Metric')
```

```


%% Attainable Maximun Likelihood (AML)
sigma_AML = 0.4;

sum = 0;
for d = 1 : diff(disparity)
    sum = sum + exp(-((squeeze(C(:,:,d)) - c1) ./ (2 * sigma_AML ^ 2)));
end

C_AML = 1 ./ sum;
C_AML = C_AML ./ max(C_AML(:));
figure;imshow(C_AML);title('Attainable Maximun Likelihood')


%% Winner Margin Naive (WMNN)
sum = 0;
for d = 1 : diff(disparity)
    sum = sum + squeeze(C(:,:,d));
end

C_WMNN = (c2 - c1) ./ sum;
C_WMNN = C_WMNN ./ max(C_WMNN(:));
figure;imshow(C_WMNN);title('Winner Margin Naive')


%% Save confidence

save(['./save_confidence/' dataset '/confidences.mat'], 'C_CUR', 'C_LC', 'C_PKRN', '
    C_MMN', 'C_NLM', 'C_MLM', 'C_AML', 'C_WMNN');

imwrite(C_CUR, ['./save_confidence/' dataset '/CUR.png'] ,'png');
imwrite(C_LC, ['./save_confidence/' dataset '/LC.png'] ,'png');
imwrite(C_PKRN, ['./save_confidence/' dataset '/PKRN.png'] ,'png');
imwrite(C_NLM, ['./save_confidence/' dataset '/MMN.png'] ,'png');
imwrite(C_NLM, ['./save_confidence/' dataset '/NLM.png'] ,'png');
imwrite(C_MLM, ['./save_confidence/' dataset '/MLM.png'] ,'png');
imwrite(C_NLM, ['./save_confidence/' dataset '/AML.png'] ,'png');
imwrite(C_NLM, ['./save_confidence/' dataset '/wmnn.png'] ,'png');


%% Combination one
C_AML = C_AML ./ max(C_AML(:));
C_MLM = C_MLM ./ max(C_MLM(:));

C_comb1 = C_AML .* C_MLM;
figure;imshow(C_comb1);title('Combination C_{AML} \cdot C_{MLM}')


%% Combination two
C_PKRN = C_PKRN ./ max(C_PKRN(:));

C_comb2 = C_AML .* C_MLM .* C_PKRN;
figure;imshow(C_comb2);title('Combination C_{AML} \cdot C_{MLM} \cdot C_{PKRN}')


%% Combination three
C_NLM = C_NLM ./ max(C_NLM(:));

C_comb3 = C_NLM .* C_MLM;
figure;imshow(C_comb3);title('Combination C_{NLM} \cdot C_{MLM}')


%% save combination
save(['./save_confidence/' dataset '/confidences_combination.mat'], 'C_comb1', '
```

```
      C_comb2', 'C_comb3');

imwrite(C_comb1, ['./save_confidence/' dataset '/comb1.png'] ,'png');
imwrite(C_comb2, ['./save_confidence/' dataset '/comb2.png'] ,'png');
imwrite(C_comb3, ['./save_confidence/' dataset '/comb3.png'] ,'png');
```

A.2 left_right_confidence.m

```
% This script compute confidence with left and right cost functions

% @author: Nicola Dal Lago
% @date: 15/09/2014
% @version: 1.0


%% clear workspace
clear; close all; clc;

dataset = 'Aloe';       %it must be 'Aloe', 'Bowling2', 'Flowerpots'

path_SGM_disparity = ['../../../C/data/Images/Middlebury/' dataset '/SGM/
      disparity_sgm.dat'];
path_SGM_occluded_pixels = ['../../../C/data/Images/Middlebury/' dataset '/SGM/
      disparity_sgm.png'];
path_SGM_occluded_pixels_right = ['../../../C/data/Images/Middlebury/' dataset '/SGM/
      disparity_sgm.png_right.png'];

rows = 555;
if(strcmp(dataset, 'Aloe'))
    cols = 641;
elseif(strcmp(dataset, 'Bowling2'))
    cols = 665;
elseif(strcmp(dataset, 'Flowerpots'))
    cols = 656;
end

load(['../../../C/data/Images/Middlebury/' dataset '/SGM/cost.mat'])
load(['../../../C/data/Images/Middlebury/' dataset '/SGM/cost_right.mat'])

% load occluded pixels' maps
tmp = double(imread(path_SGM_occluded_pixels));
SGM_occluded_matrix = tmp(:,:,3); % r,g,b      255 means pixel occluded
tmp = double(imread(path_SGM_occluded_pixels_right));
SGM_occluded_matrix_right = tmp(:, :, 3); %r,g,b

% Smallest costs and indexes
[c1, I1] = min(C,[],3);
[c1_right, I1_right] = min(C_right, [], 3);

% Second smallest costs and indexes
c2 = 32767*ones(size(c1));
I2 = ones(size(c1));
for d = 1:diff(disparity)
    index = (squeeze(C(:,:,d)) >= c1);
    index = index & (squeeze(C(:,:,d)) < c2);
    index = index & abs(I1 - d) > 1;
    if any(index(:))
        tmp = squeeze(C(:,:,d));
        c2(index) = tmp(index);
        I2(index) = d;
    end
```

```
end

% load SGM disparity
id = fopen(path_SGM_disparity);
D_SGM_line = fread(id, rows * cols, 'float');
fclose(id);

D_SGM = zeros(rows, cols);
for x = 1 : 1 : rows
    for y = 1 : 1 : cols
        current_position = ((x - 1) * cols) + y;
        D_SGM(x, y) = D_SGM_line(current_position);
    end
end

%figure;imshow(D_SGM ./ max(D_SGM(:)));title('SGM')
%figure;imshow(SGM_occluded_matrix ./ max(SGM_occluded_matrix(:)));title('occluded matrix')

%% Left Right Consistency
C_LRC = ones(rows, cols);

for x = 1 : 1 : rows
    for y = 1 : 1 : cols
        if (SGM_occluded_matrix(x, y) == 255) || (SGM_occluded_matrix_right(x, y) == 255)
        %   C_LRC(x, y) = 1;  % occluded pixel
        else

            right_position = round(x - D_SGM(x, y));
            if right_position < 1
                right_position = 1;
            end
            C_LRC(x, y) = abs(c1(x, y) - c1_right(right_position, y));
        end
    end
end

C_LRC = 1 - C_LRC;
figure;imshow(C_LRC);title('Left Right Consistency')


%% left Right Difference
C_LRD = zeros(rows, cols);

for x = 1 : 1 : rows
    for y = 1 : 1 : cols
        if (SGM_occluded_matrix(x, y) == 255) || (SGM_occluded_matrix_right(x, y) == 255)
           C_LRD(x, y) = 0;  % occluded pixel
        else

            right_position = round(x - c1(x, y));
            if right_position < 1
                right_position = 1;
            end
            C_LRD(x, y) = (c2(x, y) - c1(x, y)) / abs(c1(x, y) - c1_right(right_position, y));

        end
    end
end

figure;imshow(C_LRD);title('Left Right Difference')

```

```

%% save
save(['./save_confidence/' dataset '/confidences_left_right.mat'], 'C_LRC', 'C_LRD');
imwrite(C_LRC, ['./save_confidence/' dataset '/LRC.png'] ,'png');
imwrite(C_LRD, ['./save_confidence/' dataset '/LRD.png'] ,'png');
```

A.3 result.m

```
% This script compare 5 different cost function with ground truth
% disparity. Also excludes from the comparison occluded pixels.

% @author: Nicola Dal Lago
% @date: 09/09/2014
% @version: 2.0


%% clear workspace
clear; close all; clc;

% parameters
dataset = 'Bowling2';      % it must be 'Aloe', 'Bowling2', 'Flowerpots'

path_confidence = ['../stereo/confidence/save_confidence/' dataset '/confidences.mat'
    ];
path_ground_truth = ['../../C/data/Images/Middlebury/' dataset '/disp1.png'];
path_SGM_disparity = ['../../C/data/Images/Middlebury/' dataset '/SGM/disparity_sgm.
    dat'];
path_SGM_occluded_pixels = ['../../C/data/Images/Middlebury/' dataset '/SGM/
    disparity_sgm.png'];
path_confidence_combination = ['../stereo/confidence/save_confidence/' dataset '/
    confidences_combination.mat'];

% different images have different size
rows = 555;
if(strcmp(dataset, 'Aloe'))
    cols = 641;
elseif(strcmp(dataset, 'Bowling2'))
    cols = 665;
elseif(strcmp(dataset, 'Flowerpots'))
    cols = 656;
end

number_of_costs = 6;
cost_functions = {'comb1' 'comb2' 'comb3' 'PKRN' 'MLM' 'AML'};

% load confidence
cost = zeros(rows, cols, number_of_costs);
for i = 1 : 1 : number_of_costs - 3
   cost(:, :, i) = cell2mat(struct2cell(load(path_confidence_combination, ['C_'
        cell2mat(cost_functions(i))])));
end

for i = number_of_costs - 2 : 1 : number_of_costs
    cost(:, :, i) = cell2mat(struct2cell(load(path_confidence, ['C_' cell2mat(
        cost_functions(i))])));
end

% load ground truth disparity and SGM disparity
D_GT = double(imread(path_ground_truth));
D_GT = D_GT ./ 2; % cause the half size of the image

```

```
tmp = double(imread(path_SGM_occluded_pixels));
SGM_occluded_matrix = tmp(:,:,3); % r,g,b    255 means pixel occluded

id = fopen(path_SGM_disparity);
D_SGM_line = fread(id, rows * cols, 'float');
fclose(id);

D_SGM = zeros(rows, cols);
for x = 1 : 1 : rows
    for y = 1 : 1 : cols
        current_position = ((x - 1) * cols) + y;
        D_SGM(x, y) = D_SGM_line(current_position);
    end
end

clearvars tmp D_SGM_line id % save memory


%% compute error vector
error_vector = zeros(rows * cols, 2); % error column, occluded column (0 if occluded,
      1 if not occlude)

for x = 1 : 1 : rows
    for y = 1 : 1 : cols
        current_position = ((x - 1) * cols) + y;
        error_vector(current_position, 1) = abs(D_GT(x, y) - D_SGM(x, y));

        if(SGM_occluded_matrix(x, y) == 255) || (D_GT(x, y) == 0)  % occluded pixel
            error_vector(current_position, 2) = 0;
        else                                   % not occluded pixel
            error_vector(current_position, 2) = 1;
        end
    end
end


%% order pixels according to their confidence in decreasing order
confidence_sorted = zeros(rows * cols, 3, 5); % (confidence column, error colum,
     occluded column, number of cost function)

for i = 1 : 1 : number_of_costs

    for x = 1 : 1 : rows
        for y = 1 : 1 : cols

            current_position = ((x - 1) * cols) + y;
            confidence_sorted(current_position, 1, i) = cost(x, y, i);

        end
    end

    confidence_sorted(:, 2, i) = error_vector(:, 1); % save error
    confidence_sorted(:, 3, i) = error_vector(:, 2); % save occluded column

    % sort by first column
    tmp = confidence_sorted(:, :, i);
    confidence_sorted(:, :, i) = - sortrows(- tmp, 1);
end

clearvars cost % save memory


%% order the error vector in decreasing order
error_vector = sortrows(error_vector, 1);

```

```
%% parameter of plots
step = 20; % if step = 20 increasing comparison 5% at each time


%% number of erroneous pixels (|d_gt - d| > 1) vs number of considered pixels
error_curves = zeros(step, number_of_costs + 1);

for s = 1 : 1 : step

    percent_of_pixels = 100 - ((step - s) * (100 / step)); % 5 : 10 : 15 ... 95 : 100
    number_of_considered_pixels = round(((rows * cols) / 100) * percent_of_pixels);

    for i = 1 : 1 : number_of_costs
        number_of_erroneus_pixels = 0; % with error bigger than 1
        number_of_occluded_pixels = 0;

        for j = 1 : 1 : number_of_considered_pixels % sum

            if confidence_sorted(j, 3, i) == 0
                number_of_occluded_pixels = number_of_occluded_pixels + 1;
            elseif  confidence_sorted(j, 2, i) > 1
                number_of_erroneus_pixels = number_of_erroneus_pixels + 1; % pixel
                    with bigger than 1 error
            end

        end
        error_curves(s, i) = number_of_erroneus_pixels / (number_of_considered_pixels
            - number_of_occluded_pixels);
    end

    number_of_erroneus_pixels = 0;
    number_of_occluded_pixels = 0;
    for j = 1 : 1 : number_of_considered_pixels % sum

        if error_vector(j, 2) == 0
            number_of_occluded_pixels = number_of_occluded_pixels + 1;
        elseif error_vector(j, 1) > 1
            number_of_erroneus_pixels = number_of_erroneus_pixels + 1;
        end

    end
    error_curves(s, number_of_costs + 1) = number_of_erroneus_pixels / (
        number_of_considered_pixels - number_of_occluded_pixels);

end


%% average disparity error (|d - d_gt|) vs number of considered pixels
average_error_curves = zeros(step, number_of_costs + 1);

for s = 1 : 1 : step

    percent_of_pixels = 100 - ((step - s) * (100 / step)); % 5 : 10 : 15 ... 95 : 100
    number_of_considered_pixels = round(((rows * cols) / 100) * percent_of_pixels);

    for i = 1 : 1 : number_of_costs
        number_of_occluded_pixels = 0;
        to_sum = 0;

        for j = 1 : 1 : number_of_considered_pixels

            if confidence_sorted(j, 3, i) == 0
                number_of_occluded_pixels = number_of_occluded_pixels + 1;
            else
```

```
                to_sum = to_sum + confidence_sorted(j, 2, i);
            end

        end

        average_error_curves(s, i) = to_sum / (number_of_considered_pixels -
            number_of_occluded_pixels);
    end

    number_of_occluded_pixels = 0;
    to_sum = 0;

    for j = 1 : 1 : number_of_considered_pixels
        if error_vector(j, 2) == 0
            number_of_occluded_pixels = number_of_occluded_pixels + 1;
        else
            to_sum = to_sum + error_vector(j, 1);
        end
    end

    average_error_curves(s, number_of_costs + 1) = to_sum / (
        number_of_considered_pixels - number_of_occluded_pixels);
end


%% plot
legend_string =  cell(number_of_costs + 3, 1);
for i = 1 : 1 : number_of_costs
    legend_string(i) = cost_functions(i);
end
legend_string(number_of_costs + 1) = {'GT'};
legend_string(number_of_costs + 2) = {'Location'};
legend_string(number_of_costs + 3) = {'NorthWest'};

font_size = 16;

figure('units','normalized','outerposition',[0 0 2/3 1])
plot(linspace(number_of_costs, 100, step), error_curves, '-x')
h_title = title('number of erroneous pixels (|d - d_{gt}| > 1) vs number of
    considered pixels');
set(h_title, 'FontSize', font_size);
h_xlabel = xlabel('number of higher confidence pixels (% of total)');
set(h_xlabel, 'FontSize', font_size);
h_ylabel = ylabel('number of pixels / number of erroneus pixels');
set(h_ylabel, 'FontSize', font_size);
h_legend = legend(legend_string{:});
set(h_legend, 'FontSize', font_size);
xlim([5 100])

figure('units','normalized','outerposition',[1/3 0 2/3 1])
plot(linspace(number_of_costs, 100, step), average_error_curves, '-x')
h_title = title('average disparity error (|d - d_{gt}|) vs number of considered
    pixels');
set(h_title, 'FontSize', font_size);
h_xlabel = xlabel('Number of higher confidence pixels (% of total)');
set(h_xlabel, 'FontSize', font_size);
h_ylabel = ylabel('number of pixels / number of average erroneus pixels');
set(h_ylabel, 'FontSize', font_size);
h_legend = legend(legend_string{:});
set(h_legend, 'FontSize', font_size);
xlim([5 100])
```

Bibliografia

[1] A. Fusiello, *Visione Computazionale, appunti delle lezioni*, `http://profs.sci.univr.it/~fusiello`, 2008. 2, 3

[2] D. Pfeiffer, S. Gehrig, N. Schneider, *Exploiting the Power of Stereo Confidences*, IEEE Conference on Computer Vision and Pattern Recognition, 2013. 2, 8

[3] D. Scharstein, R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*, IJCV,47(1-3):7–42, 2002. 4

[4] H. Hirschmüller, *Accurate and efficient stereo processing by semi-global matching and mutual information*, IEEE CVPR, pages 807–814, San Diego, USA, June 2005. 4

[5] X. Hu, P. Mordohai, *A Quantitative Evaluation of Confidence Measures for Stereo Vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012. 10, 17

[6] X. Hu, P. Mordohai, *Evaluation of Stereo Confidence Indoors and Outdoors*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, USA, June 2010. 7, 9

[7] D. Scharstein, C. Pal, *Learning conditional random fields for stereo*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007. 7, 12

[8] H. Hirschmüller, D. Scharstein, *Evaluation of cost functions for stereo matching*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, June 2007. 7, 12

[9] G. Bradski, *The OpenCV Library*, `http://opencv.org/`, Dr. Dobb's Journal of Software Tools, 2000. 6

[10] D. Scharstein, R. Szeliski, *High-Accuracy Stereo Depth Maps Using Structured Light*, Proc. CVPR, volume I, pages 195–202, 2003. 12

[11] R. Haeusler, R. Nair, D. Kondermann, *Ensemble Learning for Confidence Measures in Stereo Vision*. 21