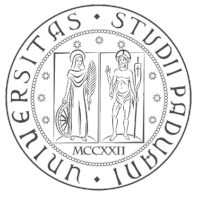


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in Ingegneria dell'Informazione

MISURE DI CONFIDENZA PER ALGORITMI DI STEREO VISION

CONFIDENCE MEASURES FOR STEREO VISION ALGORITHMS

Laureando

Nicola Dal Lago

Relatore

Prof. Pietro Zanuttigh

Correlatore

Dott. Giulio Marin

ANNO ACCADEMICO 2013/2014

Abstract

Scopo di questa tesi è il confronto di varie tecniche di stima della confidenza a partire dai dati acquisiti da una coppia di fotocamere stereo. Vengono quindi utilizzate più metriche fisse, con parametri variabili e combinazioni di esse. Si confrontano poi i risultati con la mappa di disparità *ground truth* disponibile nei dataset citati. Per il calcolo delle disparità viene usato un algoritmo di semi *Semi-global block matching (SGBM)* , opportunamente modificato, presente nelle librerie *OpenCV*. Le misure di confidenza e i risultati sono invece scritti con il linguaggio di programmazione *MATLAB*, il cui codice viene riportato nelle appendici.

Indice

Capitolo 1

Introduzione

La visione stereo è stata un'area attiva della della ricerca per decenni. Negli ultimi anni, gli algoritmi di stereo vision sono maturati a tal punto da essere applicati in un vasto scenario, dalla 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. [1]

Si possono quindi identificare due problemi: calcolo delle corrispondenze e triangolazione [1].
Il primo consiste nell'accoppiare punti delle due immagini, detti punti coniugati, che sono proiezione dello stesso punto nella scena. Il calcolo delle corrispondenze è un problema possibile in quanto le due immagini differiscono di poco, quindi un punto della scena deve apparire simile nei punti coniugati delle due immagini. Basandosi solo su questo però, sono possibili molti accoppiamenti sbagliati; le due immagini vengono quindi 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:

[1] da `https://it.wikipedia.org/wiki/Stereopsi`

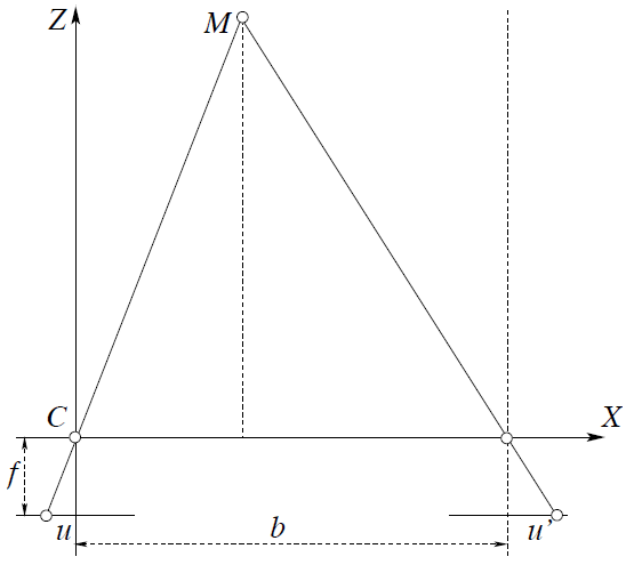


Figura 1.1: Triangolazione stereoscopica.

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

e risolvendo si ottiene:

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

dove b è la distanza tra le due fotocamere, f la focale delle fotocamere e $u' - u$ la distanza fra i due centri ottici.

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 1.2.
Gli approcci tipici per il calcolo della disparità sono basati su correlazione [3] o *semi-global matching (SGM)* [4]. In questa tesi viene utilizzato un algoritmo di tipo *SGM*, questo tipo di algoritmi usa una regione dell'immagine al posto del singolo pixel per identificare i punti coniugati. Ogni punto viene confrontato con tutti i punti nella retta epipolare, fino ad ottenere il punto coniugato e quindi la disparità. La disparità però

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

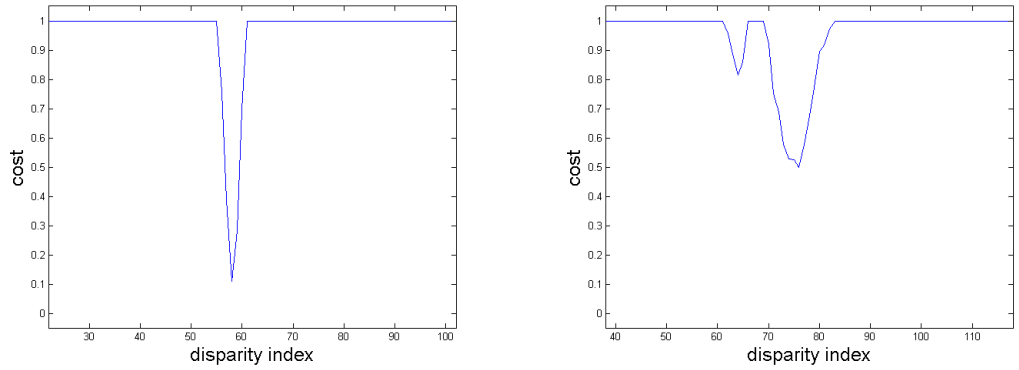
Figura 1.2: Mappa di disparità con immagine di destra come riferimento, immagine presa da `http://vision.middlebury.edu/stereo/data/`.

non è una funzione esatta, in quanto la misurazione può essere soggetta da rumore (scarsa o eccessiva illuminazione ecc...), oppure l'algoritmo di per se può sbagliare. Viene quindi generata una funzione detta "funzione di costo", la quale ha per ascissa tutti i possibili valori di disparità, e per ordinata la probabilità che quel valore di disparità sia esatto. In realtà, matematicamente parlando, non si ha una probabilità, ed è per questo che si parla di costo.

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 banale trovare l'esatta disparità. Si rende quindi necessario lo studio di varie tecniche per misurare la confidenza con la quale viene dato un determinato valore.

2.1 Implementazione utilizzata

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 di formato *.dat*, delle stesse dimensioni dell'immagine originale, ma che ha per ciascuna cella 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.

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

2.2 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 nella partita.

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

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

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

2.3 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.

Nonlinear Margin (NLM) è definito come

$$C_{NLM} = e^{\frac{c_2 - c_1}{2\sigma^2_{NLM}}} - 1 \tag{2.5}$$

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

2.4 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

$$C_{MLM} = \frac{e^{\frac{-c_1}{2\sigma^2_{MLM}}}}{\sum_d e^{\frac{c_i}{2\sigma^2_{MLM}}}} \tag{2.6}$$

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

2.5 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 la disparità con $c(x, y, d)$, ma per chiarezza chiameremo $c_R(x_R, y, d_R)$ la disparità ottenuta tenendo come riferimento l'immagine di destra. Questo genere di misure risulterà più complicata delle precedenti, in quanto prima di calcolare la disparità finale, si rende necessario calcolarla prima sia per l'immagine di destra che quella di sinistra. Per il calcolo delle due mappe di disparità, in questa tesi è stato usato l'algoritmo *Sum of Absolute Differences (SAD)*. Questo algoritmo è di tipo *block matching*, cioè calcola la mappa di disparità confrontando non il singolo pixel ma blocchi di pixel. Nel nostro caso le finestre hanno dimensione di $(2n + 1) \times (2m + 1)$

$$c_{SAD}(x, y, d) = \arg\min_d \left\{ \sum_{(k,l)} \big| I_R(x + k, y + l) - I_L(x + k + d, y + l) \big| \right\} \tag{2.7}$$

dove $k \in [-n, n]$, $l \in [-m, m]$ e $I(x, y)$ indica il livello di grigio del pixel (x, y) dell'immagine destra ($I_R(x, y)$) o sinistra ($I_L(x, y)$).

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:

$$C_{LRC}(x, y) = \big| d_1 - D_R(x - d_1, y) \big| \tag{2.8}$$

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

$$C_{LRD}(x,y) = \frac{c_2 - c_1}{\left|c_1 - \min\{c_R(x - d_1, y, d_R)\}\right|} \tag{2.9}$$

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 non 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

Anche questo tipo di misura sarà calcolata usando prima le mappe di disparità *SAD*.

Capitolo 3

Risultati

Tutte le misure effettuate vengono confrontate con una mappa di disparità chiamata *ground truth*. 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].

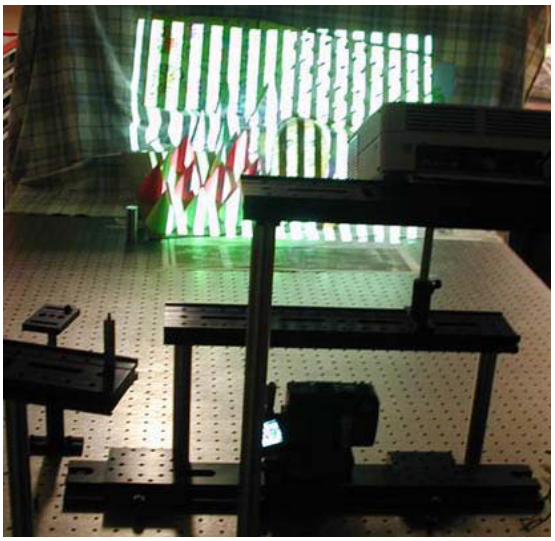


Figura 3.1: Configurazione del proiettore e della coppia di videocamere per il calcolo della disparità ground truth; si può notare come il proiettore illumina la scena con dei pattern diversi per facilitare il compito delle videocamere.

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.
Per valutare la capacità delle misure di confidenza di predire dove una disparità è corretta, è stato creato uno script *MATLAB* che ordina prima tutte le disparità ottenute

in ordine decrescente di confidenza, e poi ne calcola l'errore con densità crescente. Quindi, per ogni misura di confidenza ordinata secondo il valore di disparità, viene selezionato il primo 5% e si calcola l'errore medio in due modi:

1. Nel primo metodo si misura la percentuale di pixel che ha un errore di disparità maggiore di uno;

2. Nel secondo si misura semplicemente l'errore medio per ogni pixel.

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* e *Maximum Likelihood metric*.

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 migliore distribuzione per $\gamma = 0.9$. Per quanto riguarda l'errore medio invece, si ottengono risultati lievemente migliori per valori di γ più elevati. Questi però influiscono molto sulla uniformità della misura; ad esempio con $\gamma = 2$ si migliora solo di un inezia l'errore medio, ma la distribuzione apparterrà poi a un intervallo compreso tra circa $0,24$ e $0,5$. Come valore finale si è quindi deciso per un γ pari a $0,9$.

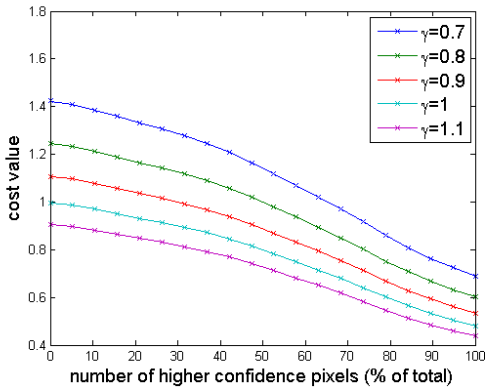


Figura 3.2: Distribuzioni diverse con valori di γ diversi.

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 minori di 1 di ϵ il costo massimo supera 1, mentre con valori troppo grandi si ha poca variazione tra il costo massimo e minimo. Il caso migliore si ha per un ϵ compreso tra $1,1$ e $1,2$, per semplicità scegliamo il valore $1,1$.

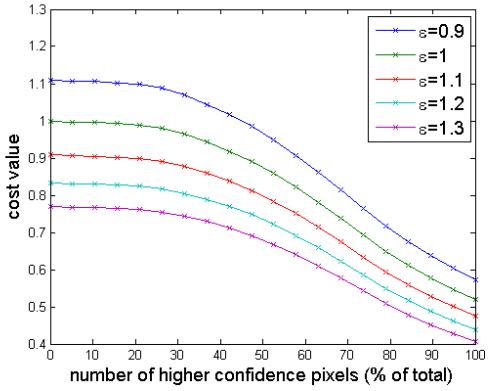


Figura 3.3: Distribuzioni diverse con valori di ϵ diversi.

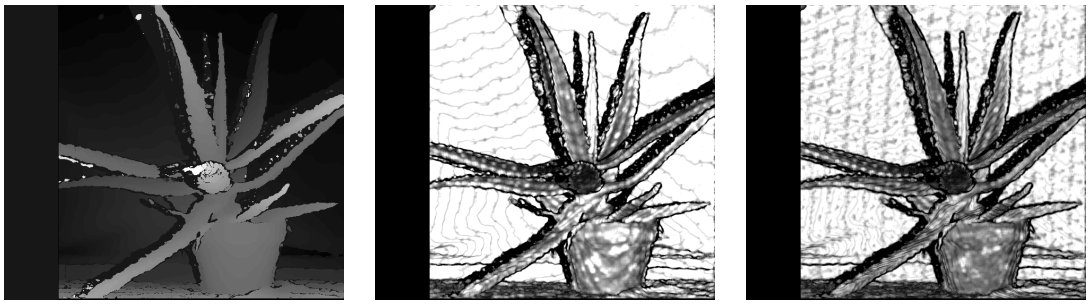
Per quanto riguardagli errori rispetto la *ground truth*, i valori per il quale e_d è maggiore di uno non subiscono praticamente nessun cambiamento al variare di epsilon. Si hanno invece piccole variazioni per quanto riguarda l'errore medio, migliorando per ϵ grandi, ma non tali da giustificare un tale peggioramento della distribuzione fra 0 e 1. Il valore finale resta $1,1$.

Variazione parametro σ per *NLM* Ripetiamo ancora una volta il procedimento, i risultati sono pressoché simili a prima, 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 $\sigma_{NLM} = 0,9$.

Variazione parametro σ per *MLM* Questa ultima misura è molto più sensibile alle variazioni parametriche, i risultati migliori in questo caso si sono ottenuti con $\sigma_{MLM} = 0,25$.

3.2 Confronto delle misure

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.



(a) *Disparità calcolata con algoritmo SGM.*

(b) *Misura Curvature.*

(c) *Misura Local Curve.*

(d) *Misura Peak Ratio Naive.*

(e) *Misura Nonlinear Margin.*

(f) *Misura Maximum Likelihood metric.*

Figura 3.4: Disparità e relative misure di confidenza.

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 *SGBM*).

Appendice A

Codice MATLAB utilizzato

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

A.1 computeConfidence.m

Questo script computa la mappa di disparità per *MSM*, *CUR*, *LC*, *PKRN*, *NLM* e *MLM*.

```
% 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;

n = 5; % image number, n = [1,5]

% stereo parameters
b = 0.176802993;
f = 856.310974;

load(['../../../C/data/Images/' num2str(n) '/Camera/Stereo/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

%% Matching Score Metric (MSM)
C_MSM = -c1;
%figure;imshow(C_MSM./max(C_MSM(:)));title('Matching Score Metric')


%%  Curvature (CUR)
Im = I1 - 1; Im(Im == 0) = 1;
Ip = I1 + 1; Ip(Ip == 97) = 96;

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

for r=1:778
    for c=1:888
        Cm(r,c) = C(r,c,Im(r,c));
        Cp(r,c) = C(r,c,Ip(r,c));
    end
end

C_CUR = -2*c1 + Cm + Cp;
figure;imshow(C_CUR./max(C_CUR(:)));title('Curvature')


%% Local Curve (LC)
gamma = 480;

Im = I1 - 1; Im(Im == 0) = 1;
Ip = I1 + 1; Ip(Ip == 97) = 96;

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

for r=1:778
    for c=1:888
        Cm(r,c) = C(r,c,Im(r,c));
        Cp(r,c) = C(r,c,Ip(r,c));
    end
end

C_LC = (max(Cm,Cp)-c1)/gamma;

figure;imshow(C_LC./max(C_LC(:)));title('Local Curve')


%% Peak Ratio Naive (PKRN)
epsilon = 128;

C_PKRN = (C2+epsilon)./(c1+epsilon) - 1;
figure;imshow(C_PKRN./max(C_PKRN(:)));title('Peak Ratio Naive')

%% Nonlinear Margin (NLM)
sigma_NLM = 80;

C_NLM = exp((C2 - c1)./(2*sigma_NLM^2)) - 1;
figure;imshow(C_NLM./max(C_NLM(:)));title('Nonlinear Margin')


%% Maximum Likelihood metric (MLM)
sigma_MLM = 8;

sum = 0;
```

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

    C_MLM =exp(-c1./(2*sigma_MLM^2))./sum;
    C_MLM=C_MLM-min(C_MLM(:));   %aggiunta????????

105 figure;imshow(C_MLM./max(C_MLM(:)));title('Maximum Likelihood Metric')


    %% Combination of the three
    P_tot = C_LC .* C_PKRN .* C_MLM;
110
    figure;imshow(P_tot./max(P_tot(:)));title('Combination')


    %% Cost of Arrigo
115 P_Arrigo = log(1+abs(b*f./ (I1+disparity(1)) - b*f./(I2+disparity(1))) .* c1./C2);
    % P_Arrigo = abs((I1+disparity(1)) - (I2+disparity(1))) .* C1./C2;

    figure;imshow(P_Arrigo./max(P_Arrigo(:)));title('Arrigo')
```

A.2 leftRightConsistency.m

```
% This script uses the LRC algorithm to compute the disparity map, from the left and
% right disparitys calculated with SAD.

% @author: Nicola Dal Lago
% @date: 01/08/2014
% @version: 1.0

%%
% clear workspace
clear; close all; clc;

leftImage = 'sx_und.png';
rightImage = 'dx_und.png';
windowSize = 9;     % window size of the block, it must be a odd number
dispMin = 48;       % minimun disparity
dispMax = 144;      % maximun disparity
threshold = 2;      % threshold value, tipically 2.0

%%

% Perform SAD Correlation (Right to Left)
fprintf('Right to Left SAD..          ');
dispMapR2L = funcSADR2L(leftImage, rightImage, windowSize, dispMin, dispMax);
fprintf('[OK] \n');

% Perform SAD Correlation (Left to Right)
fprintf('Left to Right SAD..          ');
dispMapL2R = funcSADL2R(leftImage, rightImage, windowSize,dispMin , dispMax);
fprintf('[OK] \n');

figure;imshow(dispMapR2L./max(dispMapR2L(:)));title('SAD Right to left')
figure;imshow(dispMapL2R./max(dispMapL2R(:)));title('SAD Left to right')

%%
fprintf('Left Right Consistency..     ');

dispMapL2R = - dispMapL2R;

[columns,rows] = size(dispMapL2R);
dispMapLRC = zeros(columns,rows);


for(i=1 : 1 : columns)
    for(j=1 : 1 : rows)
        xl = j;
        xr = xl + dispMapL2R(i,xl);
        if (xr>rows || xr<1)
            dispMapLRC(i,j) = 0; %% occluded pixel
        else
           xlp=xr+dispMapR2L(i,xr);
            if (abs(xl-xlp)<threshold)
                dispMapLRC(i,j) = -dispMapL2R(i,j);  %% non-occluded pixel
            else
                dispMapLRC(i,j) = 0; %% occluded pixel
            end
        end
    end
end

fprintf('[OK] \n');
figure;imshow(dispMapLRC./max(dispMapLRC(:)));title('Left Right Consistency')
```

A.3 funcSADL2R.m

```
% Compute Correlation between two images using the
% similarity measure of Sum of Absolute Differences (SAD) with Left Image
% as reference.

% @author: Nicola Dal Lago
% @date: 01/08/2014
% @version: 1.0

function dispMap = funcSADL2R(leftImage, rightImage, windowSize, dispMin, dispMax)

    leftImage = rgb2gray(imread(leftImage));
    leftImage = double(leftImage);

    rightImage=rgb2gray(imread(rightImage));
    rightImage=double(rightImage);

    [columns,rows] = size(leftImage);

    dispMap = zeros(columns, rows);

    win = (windowSize - 1)/2;
    for(i=1+win : 1 : columns-win)
        for(j=1+win+dispMax : 1 : rows-win)
            prevSAD = 65532;
            temp = 0;
            bestMatchSoFar = dispMin;
            for(dispRange=-dispMin : -1 : -dispMax)
                sad = 0;
                for(a=-win : 1 : win)
                    for(b=-win : 1 : win)
                        if (j - win + dispRange > 0)
                            temp = leftImage(i+a, j+b) - rightImage(i+a, j+b+
                                dispRange);
                            if(temp < 0)
                                temp = temp *- 1;
                            end
                            sad = sad + temp;
                        end
                    end
                end
                if (prevSAD > sad)
                    prevSAD = sad;
                    bestMatchSoFar = dispRange;
                end
            end
            dispMap(i,j) = -bestMatchSoFar;
        end
    end
```

A.4 funcSADR2L.m

```
% Compute Correlation between two images using the
% similarity measure of Sum of Absolute Differences (SAD) with
% Right Image as reference.

% @author: Nicola Dal Lago
% @date: 01/08/2014
% @version: 1.0

function dispMap=funcSADR2L(leftImage, rightImage, windowSize, dispMin, dispMax)

    leftImage = rgb2gray(imread(leftImage));
    leftImage = double(leftImage);

    rightImage = rgb2gray(imread(rightImage));
    rightImage = double(rightImage);

    [columns, rows] = size(leftImage);

    dispMap = zeros(columns, rows);

    win = (windowSize - 1)/2;
    for(i=1+win : 1 : columns-win)
        for(j=1+win : 1 : rows-win-dispMax)
            prevSAD = 65532;
            temp=0;
            bestMatchSoFar = dispMin;
            for(dispRange=dispMin : 1 : dispMax)
                sad=0;
                for(a=-win : 1 : win)
                    for(b=-win : 1 : win)
                        if (j + b + dispRange <= rows)
                            temp = rightImage(i+a, j+b) - leftImage(i+a, j+b+
                                dispRange);
                            if(temp < 0)
                                temp = temp *- 1;
                            end
                            sad = sad + temp;
                        end
                    end
                end
                if (prevSAD > sad)
                    prevSAD = sad;
                    bestMatchSoFar = dispRange;
                end
            end
            dispMap(i,j) = bestMatchSoFar;
        end
    end
```

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. 3

[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. 3

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

[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] *OpenCV library 2.4.9*, `http://opencv.org/`. 6

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