

**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**



# 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*, e il codice viene riportato nelle appendici.



# Indice

<b>Abstract</b>	<b>ii</b>
<b>1 Introduzione</b>	<b>2</b>
1.1 Stereopsi . . . . .	2
1.2 Calcolo delle corrispondenze . . . . .	3
<b>2 Misure di confidenza</b>	<b>6</b>
2.1 Implementazione utilizzata . . . . .	6
2.2 Proprietà locali della curva . . . . .	7
2.3 Minimo locale della curva . . . . .	7
2.4 Interi . . . . .	8
2.5 Consistenza fra disparità di destra e sinistra . . . . .	9
<b>3 Risultati</b>	<b>10</b>
<b>4 Combinazione di misure</b>	<b>11</b>
<b>Appendici</b>	<b>12</b>
<b>A Codice MATLAB utilizzato</b>	<b>13</b>
A.1 computeConfidence.m . . . . .	13
A.2 leftRightConsistency.m . . . . .	16
A.3 funcSADL2R.m . . . . .	17
A.4 funcSADR2L.m . . . . .	18
<b>Bibliografia</b>	<b>20</b>



# Capitolo 1

## Introduzione

La visione stereo è stata un'area attiva 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.*<sup>1</sup>

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. Basando 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:

---

<sup>1</sup>da <https://it.wikipedia.org/wiki/Stereopsi>

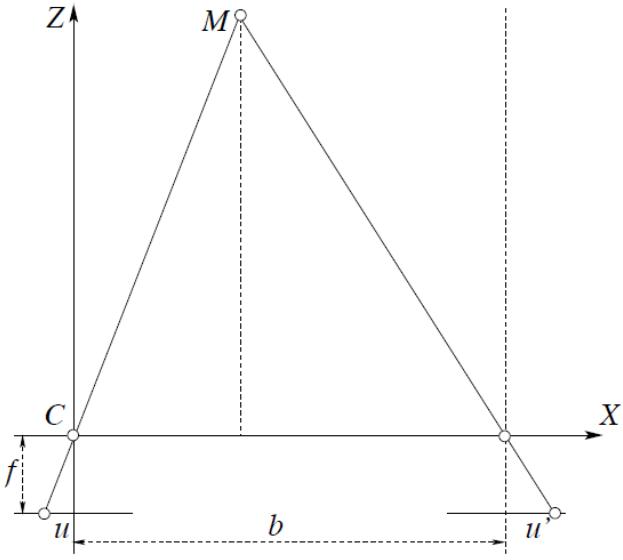


Figura 1.1: Triangolazione stereoscopica.

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

e risolvendo si ottiene:

$$z = \frac{bf}{u' - u} \quad (1.2)$$

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, 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.

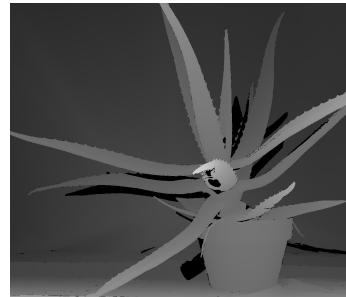
Gli approcci tipici per il calcolo della disparità sono basati su correlazione [3] o semi-global matching (SGM) [4]. In questa tesi viene utilizzato un algoritmo SGM, questo tipo di algoritmi usa una regione dell'immagine al posto del singolo pixel per identificare i punti coniugati. Ogni punto viene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



(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/>

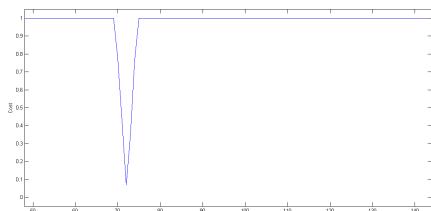
epipolare, ad ogni pixel viene dato un costo e quindi si forma una cosiddetta funzione di costo.



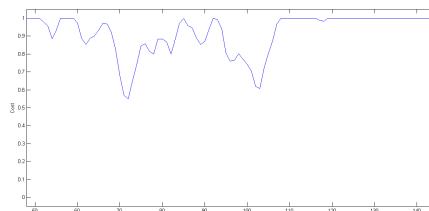
# Capitolo 2

## 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 il pixel è simile al suo possibile coniugato.



(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 il punto coniugato è individuato 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 il punto coniugato. Si rende quindi necessario lo studio di varie tecniche per misurare la confidenza di ciascun picco.

### 2.1 Implementazione utilizzata

Per il calcolo della funzione costo, è stato utilizzato un algoritmo di *semi-global matching* opportunamente modificato presente nelle librerie di *OpenCV* [9], scritto nel linguaggio *C++*. Si ottiene quindi una matrice, in formato *.dat* delle stesse dimensioni delle immagine originali, ma con la funzione costo salvata in ogni casella. Successivamente la matrice viene convertita in un formato *.mat* per renderla più facile da utilizzare con il linguaggio *MATLAB*. Poi, con uno script *MATLAB* vengono calcolate le mappe di disparità utilizzando algoritmi diversi. Le im-

magini utilizzate sono state prese dal dataset Middlebury [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

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

se  $d_1 - 1$  o  $d_1 + 1$  sono fuori dal range di disparità, il punto minimo  $c(d_1)$  viene usato due volte.

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

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

secondo il lavoro di D. Pfeiffer *e al.*[2] è stato scelto un  $\gamma$  pari a 480 per ottenere una distribuzione abbastanza omogenea.

## 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

$$C_{P KR} = \frac{c_{2m}}{c_1} \quad (2.3)$$

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

$$C_{PKRN} = \frac{c_2 + \epsilon}{c_1 + \epsilon} - 1 \quad (2.4)$$

*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;
- scegliendo un  $\epsilon = 128$ , il range dei possibili valori è limitato e la distribuzione è abbastanza uniforme tra zero e uno [2].

**Nonlinear Margin (NLM)** è definito come

$$C_{NLM} = e^{\frac{c_2 - c_1}{2\sigma_{NLM}^2}} - 1 \quad (2.5)$$

in questa tesi è stato usato un unico valore del parametro  $\sigma_{NLM}$ , senza cioè andare ad indagare i diversi risultati ottenibili con diverse varianze.

## 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

$$C_{MLM} = \frac{e^{\frac{-c_1}{2\sigma_{MLM}^2}}}{\sum_d e^{\frac{-c_i}{2\sigma_{MLM}^2}}} \quad (2.6)$$

In questo caso  $\sigma_{MLM}$  rappresenta l'incertezza della disparità, ed è stata scelta alta in modo conservativo (ad esempio  $\sigma_{MLM} = 8$ ), anche per avere una distribuzione più uniforme.

## 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  $(2n + 1) \times (2m + 1)$

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

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)** E' l'unico metodo che viene utilizzato in questa tesi per quanto riguarda la consistenza fra disparità di destra e sinistra. *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:

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

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

# 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].

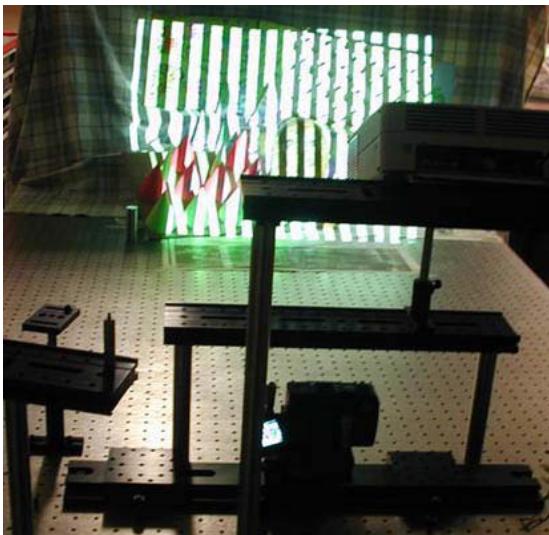


Figura 3.1: Configurazione del proiettore e della coppia di videocamere per il calcolo della disparità ground truth; si può notare come il proiettore illumina la scena con dei pattern diversi per facilitare il compito delle videocamere.

# **Capitolo 4**

## **Combinazione di misure**



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

% stereo parameters
b = 0.176802993;
15 f = 856.310974;

load(['../../../../C/data/Images/' num2str(n) '/Camera/Stereo/SGM/cost.mat'])

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

% Second smallest costs and indexes
C2 = 32767*ones(size(c1));
I2 = ones(size(c1));
25 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(:))
30     tmp = squeeze(C(:,:,d));
        C2(index) = tmp(index);
        I2(index) = d;
```

```

        end
    end
35 %% Matching Score Metric (MSM)
C_MSK = -c1;
%figure;imshow(C_MSK./max(C_MSK(:)));title('Matching Score Metric')

40 %% Curvature (CUR)
Im = I1 - 1; Im(Im == 0) = 1;
Ip = I1 + 1; Ip(Ip == 97) = 96;

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

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

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

%% Local Curve (LC)
60 gamma = 480;

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

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

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

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

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

80 %% Peak Ratio Naive (PKRN)
epsilon = 128;

C_PKRN = (C2+epsilon)./(c1+epsilon) - 1;
figure;imshow(C_PKRN./max(C_PKRN(:)));title('Peak Ratio Naive')
85

%% Nonlinear Margin (NLM)
sigma_NLM = 80;

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

%% Maximum Likelihood metric (MLM)
95 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 disparities calculated with SAD.

% @author: Nicola Dal Lago
5 % @date: 01/08/2014
% @version: 1.0

%%
% clear workspace
10 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
15 dispMin = 48;       % minimum disparity
dispMax = 144;       % maximum disparity
threshold = 2;        % threshold value, typically 2.0

%%
20 % Perform SAD Correlation (Right to Left)
fprintf('Right to Left SAD..      ');
dispMapR2L = funcSADR2L(leftImage, rightImage, windowSize, dispMin, dispMax);
fprintf('[OK] \n');

25 % Perform SAD Correlation (Left to Right)
fprintf('Left to Right SAD..      ');
dispMapL2R = funcSADL2R(leftImage, rightImage, windowSize, dispMin, dispMax);
fprintf('[OK] \n');

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

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

dispMapL2R = - dispMapL2R;

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

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

60 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.

5 % @author: Nicola Dal Lago
% @date: 01/08/2014
% @version: 1.0

10 function dispMap = funcSADL2R(leftImage , rightImage , windowSize , dispMin , dispMax)
    leftImage = rgb2gray(imread(leftImage));
    leftImage = double(leftImage);

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

    [columns ,rows] = size(leftImage);

    dispMap = zeros(columns , rows);
20
    win = (windowSize - 1)/2;
    for(i=1+win : 1 : columns-win)
        for(j=1+win+dispMax : 1 : rows-win)
            prevSAD = 65532;
25
            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;
35
                        end
                        sad = sad + temp;
                    end
                end
            end
40
            if (prevSAD > sad)
                prevSAD = sad;
                bestMatchSoFar = dispRange;
            end
        end
45
        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.

5 % @author: Nicola Dal Lago
% @date: 01/08/2014
% @version: 1.0

10 function dispMap=funcSADR2L(leftImage , rightImage , windowSize , dispMin , dispMax)
15     leftImage = rgb2gray(imread(leftImage));
    leftImage = double(leftImage);

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

    [columns , rows] = size(leftImage);

    dispMap = zeros(columns , rows);
20
    win = (windowSize - 1)/2;
    for(i=1+win : 1 : columns-win)
        for(j=1+win : 1 : rows-win-dispMax)
            prevSAD = 65532;
25
            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;
35
                            end
                            sad = sad + temp;
                        end
                    end
                end
            end
40
            if (prevSAD > sad)
                prevSAD = sad;
                bestMatchSoFar = dispRange;
            end
        end
45
        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, 7, 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.
- [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, 8
- [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, 10
- [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, 10
- [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. 10