Agenzia Nazionale per le Nuove Tecnologie,
l'Energia e lo Sviluppo Economico Sostenibile

Ministero dello Sviluppo Economico

RICERCA DI SISTEMA ELETTRICO

Realizzazione di software implementante le misure alternative dell'indice di resa cromatica

Alessandro Rizzi, Cristian Bonanomi, Saim Rasheed

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA E COMUNICAZIONE
VIA COMELICO 39 – 20135 MILANO – ITALIA

Report RdS/2011/190

REALIZZAZIONE DI SOFTWARE IMPLEMENTANTE LE MISURE ALTERNATIVE DELL'INDICE DI RESA CROMATICA

Alessandro Rizzi, Cristian Bonanomi, Saim Rasheed (Università degli Studi di Milano, Dipartimento Informatica e Comunicazione)

Settembre 2011

Report Ricerca di Sistema Elettrico
Accordo di Programma Ministero dello Sviluppo Economico – ENEA
Area: Razionalizzazione e risparmio nell'uso dell'energia elettrica
Progetto: Studi e valutazioni sull'uso razionale dell'energia:Tecnologie per il risparmio elettrico nell'illuminazione pubblica

Responsabile Progetto: Simonetta Fumagalli, ENEA

UNIVERSITÀ DEGLI STUDI DI MILANO

DIPARTIMENTO DI INFORMATICA E COMUNICAZIONE

VIA COMELICO 39 – 20135 MILANO – ITALIA

Realizzazione di software implementante le misure alternative dell'indice di resa cromatica

ALESSANDRO RIZZI • EMAIL: ALESSANDRO.RIZZI@UNIMI.IT • WWW.DTI.UNIMI.IT/rizzi

1. Introduzione

In questo report vengono presentati i software realizzati per l'implementazione delle misure alternative dell'indice di resa cromatica presentate nel report precedente. Il software è stato realizzato in Matlab (C).

2. metodi alternativi per la misura dell'indice di resa cromatica

Tutti i metodi implementati sono stati raccolti in un unico programma, per comodità.
La scelta del calcolo dei vari indici di resa cromatica, avviene attraverso la funzione:
`calculate_all_CRI(test_spectra)`
Tale funzione restituisce contemporaneamente il risultato di tutti i metodi implementati, che sono:

- `Ra_8`: indice di resa cromatica standard CIE, calcolato sulle 8 patch di Munsell, con adattamento cromatico di von Kries [1]
- `Ra_8_no_adpt`: indice di resa cromatica standard CIE, senza adattamento cromatico [1]
- `Ra_14`: indice di resa cromatica della CIE, a cui sono state aggiunte 6 patch test sature, con adattamento cromatico di von Kries [1]
- `Ra_14_no_adpt`: indice di resa cromatica modificato con l'aggiunta di 6 patch, senza adattamento cromatico [1]
- `CRI00_8`: CRI00, con 8 patch di Munsell [2, 3]
- `CRI00_10`: CRI00 con 10 patch dal Macbeth Color Checker [2, 3]
- `R96a`: nuovo tentativo della CIE (1999) [4]
- `R96_TCC_LAB`: nuovo tentativo della CIE (1999) [4]
- `eido_index_v2`: versione preliminare basata su modelli di apparenza cromatica

La funzione vuole in input lo spettro della sorgente di luce di cui si vuole calcolare il cri: `test_spectra`
Vengono fatte alcune operazioni per verificare la consistenza dei dati. Lo spettro in input infatti può avere formato diverso a seconda dello strumento utilizzato per

misurarlo. Nel caso la dimensione dello spettro sia 36, si assume che sia stato usato uno strumento che consente una misura ogni 10 nm, da 380 a 730nm (es. usando eye-one della X-Rite), quindi lo spettro viene interpolato, per averlo ogni 5 nm. Ciò viene fatto chiamando la funzione: `test_spectra = interpolate_spectra(test_spectra);`

Altrimenti viene chiesto in input quali dimensioni le dimensioni dello spettro, in particolare l'intervallo di lunghezze d'onda da considerare (es.: da 380 a 780 ogni 5 nm, questo è l'intervallo massimo possibile):

```
if (length(test_spectra)==36)
    test_spectra = interpolate_spectra( test_spectra );
    min_lambda=380;
    max_lambda=730;
    step=5;
else
    min_lambda=input('minimum lambda: ');
    max_lambda=input('maximum lambda: ');
    step=input('step of measurement (5 or 10 nm): ');
end
```

A questo punto vengono caricati tutti i dati necessari ai calcoli con la funzione `load_the_data_CRI;`

In questo modo vengono dichiarate una serie di variabili globali, tra cui le color matching functions, gli spettri delle test patch di Munsell e del Macbeth Color Checker, utilizzate nel seguito. Per eseguire il programma, è necessario copiare la cartella "valori" in C:\. Eventuali cambiamenti dei percorsi di sistema, per caricare i dati, vanno effettuati in questa funzione. Vengono poi chiamate le varie funzioni per calcolare i differenti indici di resa cromatica:

```
% calculate the standard CIE CRI, with 8 patches or 14,
with/without adapation
[Ra_8,Ra_8_no_adpt, Ra_14, Ra_14_no_adpt] =
calculate_CIE_CRI(test_spectra);

% calculate CRI00 with 8 Munsell patches
```

```
[CRI00_8] = cri00_CIE( test_spectra);

% calculate cri00 on 10 macbeth color samples
[CRI00_10] = cri00_mac( test_spectra);

% calculate Ra96a
[R96a] = calculate_R96a( test_spectra);

% calculate Ra96a TCC-Lab
[R96_TCC_LAB] = calculate_R96_TCC_LAB( test_spectra);

% new proposal
[eido_index_v1]=calculate_eido_index_v1(test_spectra);
```

Infine, vengono visualizzati a monitor i risultati.
Vediamo adesso in dettagli i vari programmi implementati.

2.1 **calculate_CIE_CRI(test_spectra)**

In questa funzione vengono calcolati gli indici di resa cromatica della CIE, con le standard 8 patch, o con le 14 patch, con o senza adattamento cromatico. La funzione esegue i passaggi già descritti nei report precedenti.
A partire dallo spettro della sorgente test viene calcolato il valore di tristimolo XYZ della sorgente test:

```
XYZ_t=spectra_light_2_XYZ(test_spectra);
```

Si passa poi alle coordinate (u,v) e quindi alle coordinate (c,d):

```
uv_t=XYZ_2_uv(XYZ_t);
cd_t=uv_2_cd(uv_t);
```

Tali valori verranno utilizzati in seguito. Sempre a partire dai valori di tristimolo della sorgente test viene calcolata la temperatura colore correlata (CCT) utilizzando il metodo di Robertson [5].

Conoscendo la temperature colore della sorgente test è possibile selezionare la sorgente di riferimento, che può essere la radiazione del corpo nero nel caso la CCT sia minore di 5000 K, oppure dedotta dalla serie Daylight nel caso sia maggiore di 5000 K:

```
if (CCT_t>=5000)
    % find the ref between the D serie
    ref_spectra=select_from_D(CCT_t, XYZ_t);
else
    % use the plank formula
    ref_spectra=spectra_from_plank(CCT_t);
end
```

A partire dallo spettro della sorgente di riferimento, si utilizzano le funzioni citate prime per passare in XYZ, poi in coordinate (u,v) e quindi (c,d).
Per ogni campione (8 o 14 test samples) è ora possibile calcolare il CRI singolo, con o senza adattamento:

```
% for every sample
for j=1:ns
    sample_spectra=test_sample_CIE_CRI(:,j);

    XYZ_s_r(j,:)=spectra_sample_2_XYZ(ref_spectra, sample_spectra);
    XYZ_s_t(j,:)=spectra_sample_2_XYZ(test_spectra,
sample_spectra);

    uv_s_r(j,:)=XYZ_2_uv(XYZ_s_r(j,:));
    cd_s_r(j,:)=uv_2_cd(uv_s_r(j,:));

    uv_s_t(j,:)=XYZ_2_uv(XYZ_s_t(j,:));
    cd_s_t(j,:)=uv_2_cd(uv_s_t(j,:));

************************************************************
% without adaptation
************************************************************
```

```
    % for the ref light
    WUV_r_no_adapt(j,:) = Yuv_2_WUV(XYZ_s_r(j,2),
uv_s_r(j,:),uv_r);

    % for the test light :
    WUV_t_no_adapt(j,:) = Yuv_2_WUV(XYZ_s_t(j,2),
uv_s_t(j,:),uv_t);

    % CRI for the single test patch

    dE(j)= deltaE_CIE_CRI(WUV_r_no_adapt(j,:),WUV_t_no_adapt(j,:));
    R_8_no_adapt(j)=100 - 4.6 * dE(j);
    R_14_no_adapt(j) = 100 - 3.54 * dE(j);

************************************************************
% chromatic adaptation
************************************************************
    % c:
    cd_s_dash_t(j,1)=cd_s_t(j,1)*cd_r(1)/cd_t(1);
    % d:
    cd_s_dash_t(j,2)=cd_s_t(j,2)*cd_r(2)/cd_t(2);

    % back to the uv space: (after the cromatic adaptation)
    uv_s_dash_t (j,:) = cd_2_uv( cd_s_dash_t(j,:) );

    % for the ref light
    WUV_r(j,:) = Yuv_2_WUV(XYZ_s_r(j,2), uv_s_r(j,:), uv_r);

    % for the test light
    WUV_t(j,:) = Yuv_2_WUV(XYZ_s_t(j,2), uv_s_dash_t(j,:),uv_r);

    % CRI for the single test patch
    dE(j)= deltaE_CIE_CRI( WUV_r(j,:), WUV_t(j,:) );
    R_8(j) = 100 - 4.6 * dE(j);
    R_14(j) = 100 - 3.54 * dE(j);
end
```

L’indice di resa cromatica standard, viene calcolato per ogni patch con la formula:

R=100 - 4.6*DE

La costante 4.6 è stata scelta affinché l’illuminante fluorescente F4 abbia un indice pari a 51. Per avere una scala confrontabile, nel caso in cui si utilizzino 14 patch invece che 8, la costante è stata modificata in 3.54, in modo che anche in questo caso F4 abbia indice di resa cromatica pari a 51.

2.2 **cri00_CIE(testlight)**

Questa funzione restituisce `CRI00_8` cioè il calcolo del CRI00 sulle 8 patch di Munsell. Questa funzione è basata sulla funzione implementata da Geisler-Moroder [2] [3].
Il workflow è simile al precedente; in questo caso la temperatura colore viene calcolata tramite la funzione:
`[temp,dist,success_cct] = cct(chrom_x_test,chrom_y_test);`
descritta nel seguito:

```
function [temp,dist,success_cct] = cct( chrom_x,chrom_y)
  [u,v] = chrom2CIE1960ucs(chrom_x,chrom_y);
  [temp,dist] = fminbnd(@(t)
cct_min_func(t,u,v),1000.0,50000.0,optimset('TolX',1e-6));
  if (dist > 0.05)
      success_cct = -1;
  else
      success_cct = 0;
  end;
```

invece che con il metodo di Robertson. In eventuali versioni successive del software verrà scelto e utilizzato uno di questi metodi.
Una volta calcolata la CCT, si computa lo spettro dell’illuminante di riferimento, come radiazione di Planck o Daylight:

```
% set reference illuminant
if (temp > 25000.0)
    disp('Error in cri00: CCT too high! (>25000K)');
    CRI00_8 = 999;
    return;
elseif (temp <= 5000.0)
    energy = planckian(temp);
else
    energy = daylight(temp);
end;
```

Per ognuno degli 8 campioni di colore viene fatto l'adattamento di Bradford [5]:

```
% apply linearized Bradford transformation for chromatic adaptation
[color_test_adapted_x,color_test_adapted_y,color_test_adapted_z] =
bradford(color_test_x,color_test_y,color_test_z,x_test,y_test,z_tes
t,x_ref,y_ref,z_ref);
```

Successivamente i valori di tristimolo sono trasformati nello spazio CIELAB, e quindi viene calcolata la funzione `ciede2000deltaE` tra il campione visto sotto la luce test e il campione visto sotto la luce di riferimento ed infine viene calcolato il CRI singolo: `CRI00i(i) = 100 - 9.097 * ciede2000_difference;`

Anche in questo caso la costante varia, per far in modo che l'illuminante F4 abbia indice CRI vicino a 51.
Il valore finale viene calcolato come media degli 8 CRI singoli:

```
CRI00_8 = round(mean(CRI00i(1:8)));
```

2.3 **cri00_mac(testlight)**

La funzione restituisce `CRI00_10` cioè il calcolo del CRI00 sulle 10 patch del Macbeth Color Checker.
Questa funzione è essenzialmente uguale alla precedente, a parte il fatto che i calcoli vengono fatti su 10 campioni test invece che 8. Nel calcolo dell'indice CRI singolo l'equazione diventa:

```
CRI00i(i) = 100 - 6.927 * ciede2000_difference;
```

2.4 **calculate_R96a(testlight)**

Nel 1999 nei commenti finali alla chiusura della commissione tecnica CIE TC 1-33, vengono proposti due indici di resa cromatica denominati R96a e R96(TCC/LAB)a.
Entrambi i metodi utilizzano 10 patch selezionate dal Macbeth Color Checker.
I due metodi differiscono sostanzialmente nel modo in cui viene scelta la sorgente di riferimento, e la costante nel calcolo finale.
Nel caso di R96a la sorgente di riferimento è scelta, a seconda della temperatura colore della sorgente test, tra una delle seguenti 6:
Daylight D65, D50, o radiazione del corpo nero alle temperature 4200 K, 3450 K, 2950 K, 2700 K. Nel caso di R96(TCC/LAB) invece si utilizza lo stesso metodo che si usa nel calcolo del CRI standard.
A tal fine, alla chiamata della funzione `load_the_data_CRI` si caricano gli spettri di queste 6 illuminanti nella variabile:

```
global d65_d50_4200_3450_2950_2700_lambda
```

L'adattamento cromatico viene fatto utilizzando la trasformazione introdotta dalla CIE nel 1994. In Matlab tale operazione è implementata nella funzione CIECAT94.

Il calcolo di ogni singolo CRI, per ogni campione test si calcola con:

```
R96a_i(i) = 100 - 3.248 * deltaE;
```

2.5 **calculate_R96_TCC_LAB(testlight)**

Le uniche differenze con la funzione precedente sono nel modo in cui viene calcolata la sorgente di riferimento, uguale al metodo standard:

```
% set reference illuminant
if (temp > 25000.0)
    disp('Error: CCT too high! (>25000K)');
return;
elseif (temp <= 5000.0)
    energy = planckian(temp);
else
    energy = daylight(temp);
end;
```

e il calcolo del singolo CRI:

```
R96_TCC_i(i) = 100 - 3.032 * deltaE;
```

2.5 **calculate_eido_index_v2(testlight)**

Questa funzione implementa una versione preliminare della nostra proposta. L'idea di base è quella di inserire un modello di apparenza cromatica che tenga conto del contesto della scena osservata. A questo scopo proponiamo l'utilizzo di un modello di computazione spaziale del colore [7], e per questa prima proposta abbiamo scelto l'algoritmo Random Spray Retinex (RSR) [8].

RSR è una variante a campionamento stocastico della versione Browniana di Retinex. In questa versione si generano dei cammini casuali (sostituiti poi dai random spray) mediante i quali ogni pixel viene ricalcolato campionando l'immagine alla ricerca del massimo locale. Questo permette un filtraggio locale dell'immagine che, a differenza delle varie trasformate utilizzate finora nei vari CRI, tiene conto dell'arrangiamento spaziale della scena considerata.

In questa prima versione la scena considerata è quella del Macbeth Color Checker (MCC) visualizzato all'interno del light boot, quindi con una distribuzione

praticamente uniforme dell'illuminante su ogni patch. Come input all'algoritmo RSR è stata costruita una immagine sintetica del Macbeth MCC. Conoscendo lo spettro di ogni patch del MCC, e lo spettro delle due illuminanti (test e riferimento), si costruiscono due immagini spettrali del MCC osservate sotto le due differenti illuminanti. A questo punto è possibile trasformare lo spettro di ogni punto dell'immagine in un valore di tristimolo XYZ e quindi in RGB.
Il risultato consiste nell'avere due immagini RGB sintetiche del Macbeth Color Checker, che possono essere date in input all'algoritmo di computazione spaziale. Questo viene fatto attraverso la funzione:

```
[img_retinex_testimg_retinex_ref]=rsr_doppio(img_rgb_test,img_rgb_ref);
```

Questo modo di procedere sostituisce la fase di trasformazione cromatica (von Kries, Bradford o CAT94), con un metodo che prende in considerazione anche il contesto spaziale della scena considerata.
Gli output dell'algoritmo sono due immagini RGB filtrate, che possono essere ritrasformate in XYZ e poi nello spazio CIELAB dove è possibile calcolare la differenza cromatica di ogni pixel usando la formula CIEDE2000.

2.6 **Coefficienti di penalizzazione**

Allo scopo di penalizzare le temperature colore estreme è stata implementata una funzione:

```
fattore_k=penalizza_k(temp);
```

descritta nel seguito:

```
function fattore_k=penalizza_k(CCT)

% in base alla CCT penalizza l'indice di resa cromatica

if ((CCT>=4000) && (CCT<=8000))
    fattore_k=1;
```

```
end

if (CCT<4000)
    % input: temperature colore
    i_start=1000;
    i_end=4000;
    % output, un valore tra 0.85 e 1
    o_start=0.85;
    o_end=1;
    fattore_k = map2range(CCT,i_start,i_end,o_start,o_end);
end

if (CCT>8000)
    % input: temperature colore
    i_start=8000;
    i_end=24000;
    % output, un valore tra 0.8 e 1
    o_start=1;
    o_end=0.8;
    fattore_k = map2range(CCT,i_start,i_end,o_start,o_end);
end
```

Le costanti di penalizzazione e il range di temperature da penalizzare possono essere scelti in base ad esperimenti effettuati con osservatori umani.

In output, oltre all'indice di resa `eido_index`, viene restituita la variabile `eido_index_no_pena` in cui si tiene conto della penalizzazione delle sorgenti a temperatura colore estrema.

Infine, il calcolo del singolo CRI per ogni patch si basa sull'equazione:

```
eido_index_i(i) = 100 - 7 * ciede2000_difference(i);
```

In questo caso la costante 7 è stata scelta per avere consistenza con la maggior parte degli altri CRI, per i quali l'indice resa cromatica dell'illuminante F4 vale 51.

3. Conclusioni

Abbiamo presentato l'implementazione di alcune funzioni per il calcolo di differenti indici di resa cromatica. In particolare: indice di resa cromatica standard (con 8-14 patch, con/senza adattamento cromatico), CRI00, R96a, R96 (TCC/LAB), ed una nuova proposta preliminare, in cui viene utilizzato un algoritmo di computazione spaziale per svolgere la fase di adattamento cromatico. Inoltre il numero di patch da utilizzare viene esteso a 24 (la totalità delle patch del Macbeth Color Checker), e include il calcolo di un fattore di penalizzazione per le temperature di colore estreme. Altre modifiche potranno essere fatte in futuro, tenendo in considerazione ulteriori test, basati sulla percezione umana.

Bibliografia

[1] Shanda J., Colorimetry, Understanding the CIE System, Wiley, 2007.

[2] D. Geisler-Moroder and A. Dur, "Color-rendering indices in global illumination methods," Journal of Electronic Imaging, vol. 18, 2009, pp. 043015-12.

[3] http://www.uibk.ac.at/mathematik/personal/geisler-moroder/

[4] Commission Internationale de l'Eclairage, Colour Rendering, TC 1-33 closing remarks, CIE Pubbl. No. 135/2, 1999.

[5] Robertson A. R., "Computation of Correlated Color Temperature and Distribution Temperature". JOSA 58, 1968.

[6] K.M. Lam, “Metamerism and Color Constancy”, Ph.D. Thesis, University of Bradford, 1985.

[7] A. Rizzi, J.J. McCann, “On the behavior of spatial models of color”, *IS&T/SPIE Electronic Imaging 2007,* S.Josè (California – USA), 28 gennaio – 1 febbraio 2007.

[8] Provenzi, E.; Fierro, M.; Rizzi, A.; De Carli, L.; Gadia, D.; Marini, D., *Random Spray Retinex: a new Retinex implementation to investigate the local properties of the model*, IEEE Transactions on Image Processing, Vol. 16, Issue 1, pp. 162-171, 2007.