

# ACE: equalizzazione cromatica per algoritmi di computer vision

Carlo Gatta, Alessandro Rizzi, Daniele Marini\*

*Dip. Tecnologie dell'Informazione – Università di Milano*  
*Via Bramante, 65 – 26013 Crema, Tel: +39 0737898089*  
*E-mail: [gatta@dti.unimi.it](mailto:gatta@dti.unimi.it), [rizzi@dti.unimi.it](mailto:rizzi@dti.unimi.it)*

*\* Dip. Scienze dell'Informazione – Università di Milano*  
*Via Comelico, 39 – 20100 Milano, Tel: +39 0737898089*  
*E-mail: [marini@dsi.unimi.it](mailto:marini@dsi.unimi.it)*

**Abstract.** Nelle applicazioni della *computer vision* può essere molto vantaggioso riprodurre alcuni processi di adattamento alle condizioni nelle quali un sistema di visione si trova ad agire. In particolare, nella robotica avanzata e nell'intelligenza artificiale applicata all'*imaging*, il contenuto informativo di un'immagine è usato per vari compiti fra cui prendere decisioni o autolocalizzare un robot in uno spazio. In questi casi vengono solitamente utilizzati il colore ed i contorni. E' noto che il colore acquisito da un dispositivo è fortemente influenzato dalle condizioni di illuminazione sia per quel che riguarda un eventuale dominante cromatica sia per la possibile presenza di configurazioni di illuminazione problematiche per un'analisi automatica dell'immagine (ad esempio controluce o illuminazione fortemente non uniforme). Per questi motivi può essere di particolare utilità disporre di meccanismi di adattamento automatico alle condizioni d'illuminazione ambientale. Funzionalità di questo tipo sarebbero in grado di aumentare, in molte occasioni, l'efficacia degli algoritmi di visione. Con questo obiettivo abbiamo sviluppato un algoritmo di equalizzazione cromatica totalmente automatico che si ispira al funzionamento del sistema visivo umano, chiamato ACE per "Automatic Color Equalization". Uno degli aspetti positivi di ACE è la massimizzazione della gamma dinamica dell'immagine, caratteristica in grado di aumentare l'efficacia di algoritmi per la segmentazione di immagini. Inoltre ACE diminuisce la variabilità del colore rispetto ai cambiamenti dell'illuminante senza richiedere informazioni a priori sull'illuminazione della scena; per questa ragione può operare in modo automatico anche in ambienti non noti, con condizioni di illuminazione non prevedibili.

## 1 Introduzione

La qualità di un'immagine acquisita da un dispositivo può essere fortemente influenzata dalle condizioni di illuminazione sia per quel che riguarda un eventuale dominante cromatica sia per la possibile presenza di configurazioni di illuminazione particolari (ad esempio controluce o illuminazione fortemente non uniforme). Il sistema visivo umano (SVU) è in grado di adattarsi efficacemente alle condizioni di luce ambientali. Per questi motivi può essere di particolare utilità disporre di meccanismi di adattamento automatico alle condizioni d'illuminazione ambientale anche per immagini digitali. Funzionalità di questo tipo sarebbero in grado di aumentare, in molte occasioni, l'efficacia degli algoritmi di visione.

In questo articolo proponiamo un algoritmo, detto ACE per "Automatic Color Equalization", derivante da un modello semplificato del SVU. Lo scopo del modello proposto è quello di

simulare alcuni meccanismi del SVU nell'intento di ottenere alcune delle sue caratteristiche di adattamento. In particolare si vuole un modello in grado di eseguire i due meccanismi fondamentali di adattamento: la *lightness constancy* e la *color constancy*.

Il meccanismo di *lightness constancy*, che permette al SVU di adattarsi a diversi livelli di luminosità della scena, fa sì che il SVU percepisca come grigio medio l'oggetto che riflette la luminosità media presente nel campo visivo [1]. Dal punto di vista prettamente algoritmico questo meccanismo corrisponde all'operazione di centratura dell'istogramma attorno al grigio medio disponibile rispetto alla gamma dinamica utilizzata per rappresentare un'immagine digitale; faremo riferimento a questo meccanismo con il nome di *gray world*. Questo meccanismo globale non è di per sé di particolare efficacia [2] ma è di fondamentale importanza se inserito in un modello più completo.

Il meccanismo di *color constancy* è in grado di rimuovere una eventuale dominante cromatica dovuta al tipo di illuminante presente nell'ambiente. In questo caso il SVU modifica la propria percezione del colore adattandosi ad un'ipotetica zona bianca presente nella scena. Faremo riferimento a questo meccanismo con il nome di *white patch* [3]. Questo meccanismo opera globalmente sull'immagine; nel modello proposto, oltre a questo, come vedremo in seguito, abbiamo aggiunto un meccanismo di correzione cromatica locale.

## 2 L'implementazione del modello: ACE

Lo schema base dell'algoritmo è presentato in Fig. 1: Il primo stadio esegue un ricalcolo del valore di ogni pixel in funzione del resto dell'immagine (Adattamento Cromatico Spaziale), mentre il secondo stadio (Mappaggio della dinamica) esegue un accurato mappaggio della dinamica prodotta dal primo stadio in quella disponibile dal dispositivo di visualizzazione.

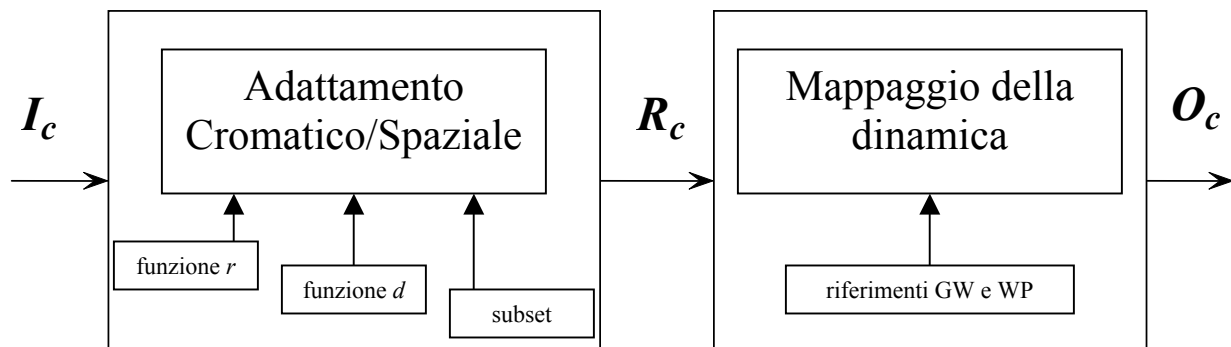


Figura 1: Schema fondamentale di ACE

L'algoritmo non richiede alcun tipo di supervisione e nessun tipo di precalcolo statistico.

Nella Fig. 1,  $I$  indica l'immagine in input,  $R$  il risultato intermedio mentre  $O$  rappresenta il risultato finale; il pedice  $c$  denota il canale cromatico (RGB) poiché l'algoritmo viene applicato separatamente sui tre canali cromatici.

## 2.1 Adattamento cromatico spaziale

Il primo stadio produce un immagine intermedia  $R$  dove ogni pixel è stato ricalcolato in funzione del contenuto cromatico dell'immagine. Ogni pixel  $p$  di  $R$  viene ottenuto eseguendo l'operazione descritta dalla equazione (1).

$$R_c(p) = \frac{\sum_{j \in \text{Subset}, j \neq p} \frac{r(I(p) - I(j))}{d(p, j)}}{\sum_{j \in \text{Subset}, j \neq p} \frac{r_{\max}}{d(p, j)}} \quad (1)$$

$I(p) - I(j)$  simula un meccanismo di inibizione laterale,  $d(p, j)$  è una funzione di distanza fra pixel che incorpora il concetto di località,  $r(\cdot)$  è la funzione, descritta in dettaglio successivamente, che calcola l'apparenza relativa fra due pixel. L'elaborazione di ogni pixel può essere fatta confrontandolo con l'intera immagine o ristretta ad un suo *subset*.

Il denominatore nell'equazione (1) è stato introdotto per bilanciare il filtraggio in funzione della posizione di  $p$  e compensare "l'effetto bordo"; omettere questa parte significa ottenere un alone ellittico nell'immagine finale poiché i pixels vicini al bordo verrebbero ricalcolati su un vicinato non omogeneamente distribuito.  $r_{\max}$  è il valore massimo assunto dalla funzione  $r(\cdot)$ .

Il meccanismo di inibizione realizzato attraverso la differenza  $I(p) - I(j)$ , unito alla funzione  $r(\cdot)$  è in grado di controllare il contrasto dell'immagine finale.

## 2.2 L'effetto locale/globale

La distanza  $d(\cdot)$  è responsabile del comportamento locale e globale dell'algoritmo proposto. E' noto che entrambi questi comportamenti sono presenti nel SVU umano. Infatti, i modelli del SVU con comportamento esclusivamente globale non sono in grado di spiegare alcuni meccanismi di modifica locale della percezione come, ad esempio, l'effetto di contrasto di simultaneità o le bande di Mach.

Durante i test preliminari sono state provate diverse funzioni  $d(\cdot)$ , ma nessuna di queste si è imposta come funzione ottimale. Sono state provate:  $r$  (distanza Euclidea in pixels),  $1 / e^{-\alpha r}$ , distanza *Manhattan*,  $r^2$ , *Manhattan*<sup>2</sup>. A parte le ultime due che danno risultati insoddisfacenti, la scelta della migliore distanza è ancora oggetto di investigazione.

Per i test effettuati in questo articolo è stata utilizzata la distanza Euclidea  $r$ , la relativa funzione peso che bilancia il comportamento locale/globale ( $1/r$ ) è mostrata in Fig. 2.

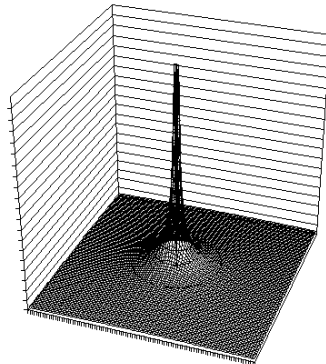


Figura 2: Rappresentazione 3D della funzione peso  $1/r$

### 2.3 La luminosità apparente relativa

Per ogni pixel dell'immagine iniziale,  $r(\cdot)$  e  $d(\cdot)$  controllano l'interazione dovuta alla differenza di luminosità su ogni canale cromatico fra pixel considerando allo stesso tempo la distribuzione spaziale del colore. L'algoritmo somma ogni singolo contributo fornito da ogni confronto pixel-pixel pesandolo tramite la funzione distanza  $d(\cdot)$  per giungere al valore finale di ogni pixel.

Per ottenere un comportamento *gray world*  $r(\cdot)$  deve essere una funzione pari mentre il comportamento *white patch* è ottenuto grazie all'amplificazione non lineare delle differenze di luminosità fra pixel.

Abbiamo provato diverse funzioni  $r(\cdot)$  con l'obiettivo di implementare un comportamento *white patch* efficace. Nella Tabella 1 sono mostrate le varianti implementate con eventuali parametri mentre in Fig. 3 sono mostrati i grafici delle rispettive funzioni.

Tabella 1: Funzioni  $r(\cdot)$  testate

Tipo	Parametri	Range
Linear	-	-
Signum	-	-
Saturation	pendenza	$[1, \infty)$

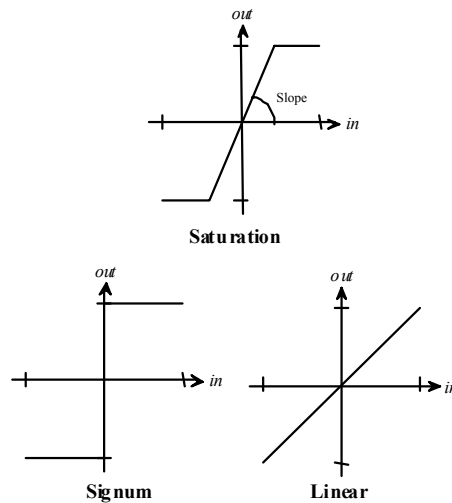


Figura 3: Grafici delle funzioni  $r(\cdot)$  testate

Le funzioni *Linear* and *Signum* possono essere pensate come casi limite della funzione *Saturation* con pendenza 1 e infinito rispettivamente. I test in questo articolo sono stati eseguiti con la funzione *Saturation* con pendenza 20.

### 2.4 Mappaggio della dinamica

Il secondo stadio di Fig. 1 mappa i valori della matrice intermedia  $R$  in quelli dell'immagine finale  $O$ . In questo stadio non solo si può utilizzare tutta la dinamica disponibile di  $O$  (mappaggio lineare classico), ma si possono considerare alcuni valori di riferimento per ottenere un mappaggio più accurato.

Come conseguenza dei valori di riferimento scelti, il secondo stadio può aggiungere un ulteriore adattamento *gray world* e *white patch* a livello globale.

I seguenti due metodi sono stati implementati per ottenere in uscita delle immagini standard a 24 bit partendo dalla matrice di valori floating point  $R$ .

#### 2.4.1 Mappaggio lineare classico

Questo semplice metodo scala linearmente il range di valori  $R_c$  indipendentemente su ogni canale cromatico  $c$  nella gamma  $[0,255]$  utilizzando la seguente formula:

$$O_c(p) = \text{round}[127.5 + s_c R_c(p)] \quad (2)$$

dove  $s_c$  è la pendenza del segmento  $[(m_c, 0), (M_c, 255)]$ , con

$$M_c = \max_p [R_c(p)] \quad (3)$$

e

$$m_c = \min_p [R_c(p)] \quad (4)$$

In questo caso il mappaggio lineare classico ricopre esattamente l'intera dinamica disponibile senza ulteriori accorgimenti.

#### 2.4.2 Mappaggio White Patch / Gray World

Questo metodo alternativo fornisce migliori risultati scalando linearmente i valori in  $R_c$  con la formula (2) ma utilizzando  $M_c$  come "bianco" di riferimento per ogni canale cromatico e lo zero ottenuto in  $R_c$  come stima del "grigio medio". Per questo motivo non è detto che tutta la dinamica disponibile sia utilizzata interamente facendo perdere alcuni toni scuri; è inoltre possibile che alcuni valori siano mappati in valori negativi, in questo caso i valori negativi vengono impostati a zero.

Il secondo metodo presentato aggiunge un ultimo adattamento *gray world* globale, in questo modo tutte le immagini risultanti hanno una distribuzione centrata attorno al grigio medio.

### 3 Complessità algoritmica e metodi di velocizzazione per il real-time

L'algoritmo presentato ha complessità algoritmica  $O(N^2)$  con  $N$  numero dei pixel dell'immagine. Per questo motivo è impensabile il suo utilizzo per la robotica senza una sua velocizzazione. A questo riguardo sono stati già sviluppati diversi approcci [4]. Alcuni di questi come ad esempio il metodo Look Up Table (LUT) sono di facile implementazione ma tolgono le caratteristiche di filtraggio locale di ACE. A questo scopo è stata sviluppata una variante al metodo LUT in grado di mantenere le caratteristiche locali del filtraggio. Il metodo sviluppato si chiama Local Linear LUT (LLLUT) poiché prevede l'utilizzo di LUT locali derivanti da una interpolazione lineare. La descrizione completa del metodo (peraltro utilizzabile con qualsiasi filtro per immagini di cui si voglia conservare l'effetto locale) si può trovare in [5].

Il grande vantaggio di questo approccio è che si possono ottenere risultati molto simili a quelli ottenibili con il filtraggio con ACE, ma con tempi di computazione drasticamente ridotti; se si imposta un fattore di decimazione  $d$  (ogni asse scalato per  $d$ ) e se il tempo di computazione con ACE è  $t_{ACE}$ , il tempo necessario per la computazione di ACE con il metodo LLLUT si riduce a:

$$t_{LLLUT} = \frac{t_{ACE}}{d^4}.$$

Per fare un esempio, il filtraggio di un'immagine 300x240 (72000 pixels) con ACE su un Pentium-III™@450 con Windows2000™ richiede 28'18'' mentre con l'utilizzo della tecnica LLLUT con la stessa configurazione e fattore di decimazione  $d=4$  richiede 6'' e con  $d=6$  meno di 2''. Il tempo di computazione diminuisce drasticamente mentre l'accuratezza del filtraggio rimane sostanzialmente costante [5].

#### 4 ACE per la visione artificiale: risultati

Di seguito mostriamo due situazioni in cui un algoritmo di equalizzazione automatica può essere di aiuto per la visione artificiale.

##### 4.1 Prefiltraggio per algoritmi di riconoscimento basati sul colore

L'acquisizione di una immagine digitale può essere fortemente influenzata dall'illuminante della scena. In genere, i dispositivi di acquisizione sono in grado di adattarsi facilmente ai livelli di luminosità della scena ma con più difficoltà alle variazioni cromatiche dell'illuminante. Inoltre l'adattamento automatico al livello di luminosità è solitamente una funzione applicata allo stesso modo sull'intera immagine e questo rende impossibile un qualsiasi adattamento efficace nel caso di forti disparità di illuminazione nella scena (ad esempio in caso di controluce).

Ipotizziamo che un robot utilizzi informazioni cromatiche per auto-localizzarsi e/o per individuare oggetti e obiettivi: Cambiando le condizioni di illuminazione (cosa probabile se il robot naviga in un ambiente con diverse sorgenti di illuminazione artificiali e/o naturali) il colore acquisito può cambiare di molto e questo può influire pesantemente sulle performance dell'algoritmo di riconoscimento e/o navigazione.

Per valutare il possibile utilizzo di ACE nel risolvere problematiche di questo tipo abbiamo misurato la capacità di mantenere il colore di un oggetto il più possibile invariante rispetto all'illuminante; questo senza avere informazione alcuna sul tipo di illuminante sotto la quale si è acquisita l'immagine. Per un utilizzo di questo tipo risulta più utile una misura di *color constancy* che preveda l'invarianza del colore tra molteplici illuminanti piuttosto che rispetto ad un illuminante di riferimento.

Per misurare questa caratteristica utilizziamo la distanza cromatica  $\Delta E_{mean}$  nello spazio colore CIELa\*b\* fra due immagini calcolata come la media delle distanze fra le coppie di pixel corrispondenti nelle due immagini, come nell'equazione (5):

$$\Delta E_{mean} = \frac{\sum_{x=0}^{sizex} \sum_{y=0}^{sizey} \Delta E(I_1(x, y), I_2(x, y))}{sizex \cdot sizey} \quad (5)$$

dove  $I_1$  e  $I_2$  sono le immagini sulle quali calcolare la distanza cromatica  $\Delta E_{mean}$ .

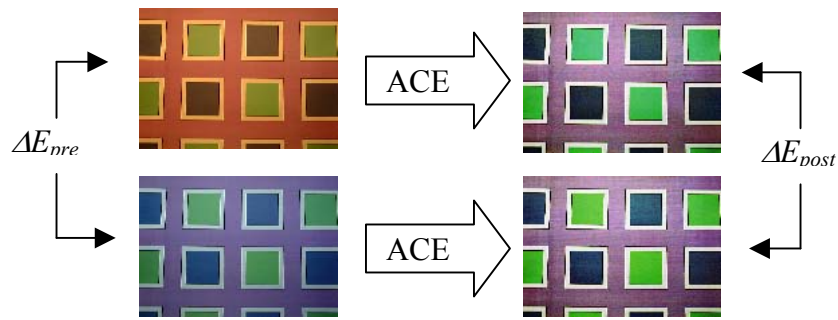


Figura 4: Procedura per la valutazione della capacità di normalizzazione cromatica di ACE

Abbiamo testato la capacità di *color constancy* di ACE su due database formati da scene e oggetti ripresi sotto diversi illuminanti. Si valuta dunque la  $\Delta E_{mean}$  prima e dopo il filtraggio con ACE. Quanto più il  $\Delta E_{mean}$  diminuisce tanto più l'algoritmo dimostra capacità di normalizzazione cromatica. La procedura è indicata schematicamente in Fig 4.

Le immagini utilizzate sono il *University of East Anglia (UEA) uncalibrated color image database* e un set di sei immagini sintetiche ottenute con un ray-tracer fotometrico partendo dalla stessa scena tridimensionale in sei diverse condizioni di illuminazione.

L' *UEA uncalibrated colour image database* è un set di 392 immagini formate da 28 texture illuminate da tre diverse luci e acquisite con 4 camere digitali (variando da macchine professionali ad amatoriali) e 2 scanner. Le immagini sono state acquisite illuminandole con una lampada A CIE, una D65 e una TL84. Per questo test non si è ritenuto necessario tutto il database, è stato deciso di utilizzare solo il set acquisito con la camera digitale Fuji Mx-700. Un esempio è mostrato in Fig. 5.

Le sei immagini di sintesi sono state ottenute con un ray-tracer fotometrico descritto in [6] da un modello 3D di una sala. Un esempio è mostrato in Fig. 6. Le luci utilizzate per la generazione sono una A, B, C, D65 standard CIE e una lampada al mercurio; l'ultima immagine è ottenuta con un misto dei precedenti illuminanti.

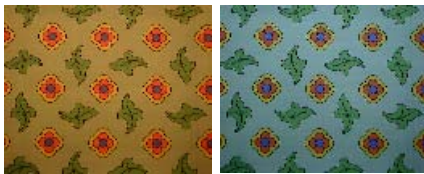


Figura 5: due immagini del IDB UEA



Figura 6: due immagine sintetiche

I risultati della procedura precedentemente descritta applicati a queste immagini sono mostrati nelle tabelle 2 e 3 dove sono rappresentati i valori di  $\Delta E$  medio fra ogni coppia di immagini prima e dopo il filtraggio con ACE. La stessa procedura è stata utilizzato per l'IDB UEA dando i risultati mostrati in tabella 4. Come si può notare i valori di  $\Delta E$  per ogni coppia di immagini decrescono dopo il filtraggio con ACE. Nel caso delle immagini sintetiche i valori decrescono in modo considerevole sotto la soglia del valore  $\Delta E=10$  che indica che i colori confrontati sono molto simili.

Tabella 2:  
 $\Delta E$  medio prima del filtraggio

Prima	A	B	C	D65	Hg	Mix
A	0					
B	36.12	0				
C	52.56	17.06	0			
D65	48.56	12.51	6.45	0		
Hg	46.1	18.42	21.05	19.15	0	
Mix	27.1	9.52	25.76	21.55	24.49	0

Tabella 3:  
 $\Delta E$  medio dopo il filtraggio

Dopo	A	B	C	D65	Hg	Mix
A	0					
B	6.19	0				
C	7.22	1.81	0			
D65	6.94	1.68	1.33	0		
Hg	8.48	5.05	5	5.09	0	
Mix	8.74	8.08	8.63	8.46	10.34	0

Tabella 4:  $\Delta E$  medio sulle 3 illuminanti per ogni immagine del IDB UEA, prima e dopo il filtraggio con ACE

N°	Prima	Dopo	Dopo/Prima
1	40,72	19,73	0,48
2	44,34	17,23	0,39
3	40,39	26,69	0,66
4	41,76	15,53	0,37
5	-	-	-
6	37,64	18,66	0,5
7	42,11	20,53	0,49
8	41,30	15,20	0,37
9	45,15	25,33	0,56
10	41,96	26	0,62
11	39,53	17,29	0,44
12	47,97	23,05	0,48
13	48,26	24,6	0,51
14	39,84	14,06	0,35

15	38,54	11,52	0,3
16	39,96	13,61	0,34
17	47,12	28,37	0,6
18	42,7	30,97	0,73
19	41,49	15,34	0,37
20	47,89	25,26	0,53
21	43,19	22,03	0,51
22	38,18	17,47	0,46
23	41,16	22,46	0,55
24	40,27	20,11	0,5
25	43,18	18,27	0,42
26	39,87	15,95	0,4
27	44,78	37,29	0,83
28	42,12	27,66	0,66
<b>Media</b>	<b>42,27</b>	<b>21,12</b>	<b>0,5</b>

Nel caso del IDB UEA le distanze vengono ridotte in media del 50% mostrando una forte capacità di filtraggio di diversi tipi di dominanti cromatiche.

#### 4.2 Prefiltraggio per algoritmi di edge-detection

La rilevazione dei contorni degli oggetti è l'elaborazione base per il riconoscimento di forme. Gli algoritmi di edge-detection si sono evoluti ricercando sempre tecniche più sofisticate dovendo spesso scontrarsi con problemi di riconoscimento di oggetti in ombra o di situazioni in cui la dominante cromatica rende difficilmente distinguibile un oggetto rispetto allo sfondo. Per mostrare i miglioramenti ottenibili mediante prefiltraggio con ACE utilizzeremo una metodologia di estrazione dei contorni molto semplice. L'utilizzo di tecniche più sofisticate non può che portare a risultati migliori.

Scelta un'immagine viene eseguito il filtraggio con ACE velocizzato con la tecnica LLLUT con il parametro  $d$  settato in modo che l'elaborazione richieda un tempo inferiore ai 2 secondi (compreso caricamento e salvataggio dell'immagine ed allocazione delle strutture dati). Sia per l'originale che per la filtrata viene effettuata la convoluzione con il kernel LoG mostrato nella Tabella 5 e successivamente viene praticata una soglia tale per cui i pixel vengono ricomputati secondo l'espressione (6). Scelti alcuni valori di soglia (ad esempio 128, oppure 192) si dà una valutazione quantitativa considerando quanti pixels in percentuale sono stati identificati come contorno con e senza l'utilizzo del prefiltraggio con ACE. Oltre a questa semplice valutazione oggettiva verranno evidenziate alcune peculiarità del prefiltraggio e i vantaggi che ne conseguono.

$$v_{out} = \begin{cases} 255 & \text{se } v_{in} > \text{soglia} \\ 0 & \text{se } v_{in} \leq \text{soglia} \end{cases} \quad (6)$$

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

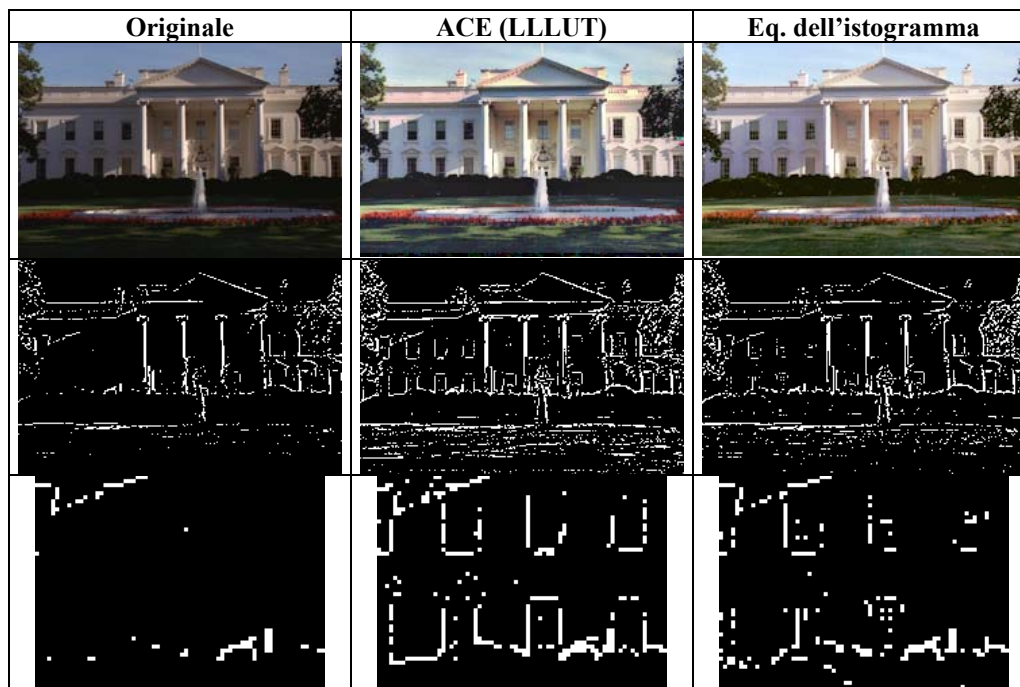
Tabella 5: Kernel di convoluzione per l'estrazione dei contorni

Nella tabella 6 è indicato il nome dell'immagine, la percentuale di pixels identificati come contorno senza e con il prefiltraggio per i valori di soglia utilizzati nell'espressione (6). Come si può notare l'utilizzo di ACE come prefiltraggio favorisce l'identificazione dei contorni con tutti i livelli di sogliatura scelti. Questo è dovuto alla capacità di ACE di massimizzare la dinamica dell'immagine. Va detto che il fatto di identificare più contorni non è di per sè necessariamente significativo. Inoltre si potrebbe pensare di massimizzare la dinamica con metodi più semplici e computazionalmente meno onerosi, come l'equalizzazione d'istogramma. Proprio per evidenziare come l'equalizzazione di ACE sia naturale e "data driven" mostriamo un risultato fra quelli applicati confrontandolo con l'originale e con l'equalizzazione d'istogramma. La tabella 7 mostra nella prima riga l'originale, il filtraggio con ACE e l'equalizzazione d'istogramma; nella seconda riga è mostrata l'immagine ottenuta con la procedura precedentemente descritta con *soglia*=224 partendo dalle rispettive immagini mentre nella terza si può notare un dettaglio ingrandito delle finestre in ombra. Dall'immagine originale non si riesce (con questo valore di soglia) ad estrarre i contorni delle finestre in ombra e ancor meno le porte centrali di ingresso. Apparentemente il filtraggio di ACE e l'equalizzazione dell'istogramma sembrano molto simili ma se confrontiamo i risultati dell'edge detection notiamo come ACE sia più efficace, permettendo di catturare contorni più regolari nelle zone in ombra. Questo è dovuto alla capacità di ACE di correggere localmente i difetti di esposizione dovuti a disparità di illuminazione in diverse zone dell'immagine.

Tabella 6: Risultati numerici del confronto con e senza prefiltraggio

Nome immagine	Soglia=128		Soglia=192		Soglia=224	
	Originale	ACE	Originale	ACE	Originale	ACE
Ferrari	5,28	9,99	3,4	6,51	2,62	5,13
Vicolo	12,49	25,2	7,52	18,6	5,89	15,41
Sintetica A	9	14,34	6,39	10,95	5,11	9,59
Caterpillar	8,25	16,89	6,2	12,59	5,19	10,91
Traghetto	9	16,3	5,27	10,45	3,82	8,49
Casa bianca	10,77	20,69	6,72	13,98	5,29	11,09

Tabella 7: Confronto visivo dei risultati ottenibili con ACE e con l'equalizzazione dell'istogramma



## 5 Conclusioni e prospettive

In questo articolo abbiamo presentato un algoritmo di equalizzazione cromatica automatica (ACE) e lo abbiamo proposto come prefiltraggio in grado di migliorare le prestazioni di algoritmi di visione automatica. ACE ha mostrato capacità di filtraggio sia locali che globali riducendo le differenze cromatiche dovute alle variazioni dell'illuminante e massimizzando la dinamica di un'immagine utilizzandone le informazioni cromatiche locali. Data la sua complessità algoritmica è stato sviluppato un metodo di velocizzazione (LLLUT) in grado di non perdere le sue caratteristiche più salienti.

Sviluppi futuri prevedono di incorporare nel modello altre caratteristiche del SVU fra le quali il meccanismo di assimilazione. Inoltre è previsto lo studio e l'implementazione di ulteriori metodi di velocizzazione.

Una versione di ACE è scaricabile dal sito internet [7].

### Bibliografia:

- [1] G. Buchsbaum, "A spatial processor model for object color perception", *J. Franklin inst.*, 1980, 310 (1), pp. 1-26.
- [2] Rizzi, C. Gatta, D. Marini, "Color Correction between Gray World and White Patch", *Electronic Imaging 2002*, 20-25/01/02, S. Josè, California (USA)
- [3] B. Funt and V. Cardei. "Committee-based color constancy", *J. Opt. Soc. Am. A*, 11 (11):3011-3020, 1994.
- [4] G. Ciocca, D. Marini, A. Rizzi, R. Schettini, and S. Zu. "Retinex preprocessing of uncalibrated images for color based image retrieval" in *CBMI2001 Second International IEEE Workshop on Content Based Multimedia and Indexing*, Brescia (Italy), 2001.
- [5] A. Rizzi, C. Gatta, "A Local Lut Method for increasing the speed of generic image filtering algorithms", technical report, Univ. of Milano, Milano (Italy), 7/2002.
- [6] D. Marini, A. Rizzi, M. Rossi, "Color constancy measurements for synthetic image generation", *Journal of Electronic Imaging*, Vol. 8, N.4, october 1999, pp.394-403.
- [7] <http://saturn.media.dsi.unimi.it/~marini/CGIV02/ace.html>.