# Universitat Politècnica de Catalunya

Visió per Computador
Grau en Enginyeria Informàtica

---

## EXERCICI 2 DE LABORATORI

## IMAGING THE ORION NEBULA

---

*Autor:*
Daniel DONATE

*Professor:*
Manuel FRIGOLA
Q2 Curs 2020-2021

3 de març del 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

# Enunciat de l'exercici

L'objectiu del problema és fusionar dues imatges de la nebulosa d'Orion en una única imatge per aconseguir un millor contrast i reducció del soroll, tal i com es suggereix a la següent figura.

*Figura 1: Nebulosa d'Orion. Imatge original (esquerra) i imatge objectiu (dreta)*

Amb aquest propòsit, avaluarem diferents procediments per introduir funcions de contrast no lineals que ens aproparan al resultat desitjat.

Recordem que l'objectiu d'aquests procediments és augmentar el contrast de les zones fosques de la imatge $Am$ per poder visualitzar més adequadament els colors de la nebulosa. Per tant, hem d'introduir funcions que *passin* amb certa *rapidesa* de valors foscos a valors amb més il·luminació, de manera que augmentem la brillantor de les zones fosques de la imatge.

# Provant la funció exponencial

Una primera possibilitat passa per l'ús d'una funció molt habitual en *l'enhancement* d'imatges, que s'aplica, lògicament, a cada píxel $x$ de la nostra imatge combinada:

$$f(x) = 1 - e^{-kx}$$

per un cert valor $k$ positiu, generalment enter. Notem que aquest valor és el que marcarà com de ràpid és aquest canvi que anunciàvem des de valors foscos a valors amb major il·luminació. Podem visualitzar com canvia la funció $f(x)$ per diferents valors de $k$ en la *Figura 2*. Notem que, com més gran és el valor de $k$, més ràpid decau la funció exponencial i més s'apropa $f(x)$ a 1. Per tant, més brusc és el contrast de la imatge resultant. Amb valors de $k$ més baixos, en canvi, tenim un comportament més semblant a la linealitat per píxels molt propers al negre. De manera que el contrast de la nova imatge serà més suau i, presumiblement, més *agradable* a l'ull, ja que l'efecte d'un contrast molt *sobtat*, que afecta a píxels molt foscos (com, per exemple, el que s'obté usant $k = 30$) enlluerna la imatge, reduint la nitidesa de la nebulosa.

A la *Figura 3* observem, d'esquerra a dreta, i de dalt a baix, el resultat d'aplicar la funció $f(x)$ sobre la imatge $Am$ amb valors de $k = 5, 10, 20, 30$, respectivament. Com anunciàvem, la *millor* imatge que obtenim és la que es correspon al valor $k = 5$, almenys en quant a similitud amb la imatge objectiu que volem simular.

La funció es pot aplicar amb la següent línia en MATLAB, utilitzant un valor fixat $k$:

```
17    Am = arrayfun(@(x) 1-exp(-k*x),Am);
```

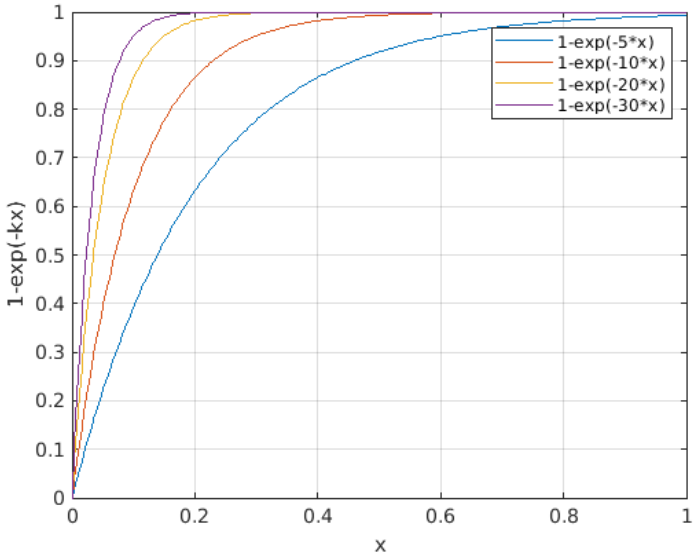


*Figura 2: Comportament de la funció de contrast exponencial per diferents valors de k*

*Figura 3: Imatges obtingudes aplicant f(x) = 1-exp(k·x) amb diferents valors de k*

## Provant la correcció gamma

La correcció gamma és un cert tipus d'operació no-lineal que s'utilitza per a codificar i decodificar luminància en sistemes de vídeo o imatge. En la seva forma més senzilla, la correcció gamma està definida per la següent llei de potències:

$$V_{out} = AV_{in}{}^{\gamma}$$

on $A$ és una constant, i les entrades i sortides són valors reals no negatius. En el cas més comú: $A = 1$ i les entrades i sortides són valors en el rang [0-1].

Aquest procediment de contrast es pot utilitzar mitjançant la funció *imadjust* de MATLAB, especificant el paràmetre de correcció, $\gamma$. Notem que, per valors de $\gamma > 1$, es produeixen ombres

més fosques, mentre que amb valors de $\gamma < 1$, provoquem que les zones més fosques es facin més clares. Aquest comportament s'il·lustra en la *Figura 4*, per diferents valors de $\gamma$.

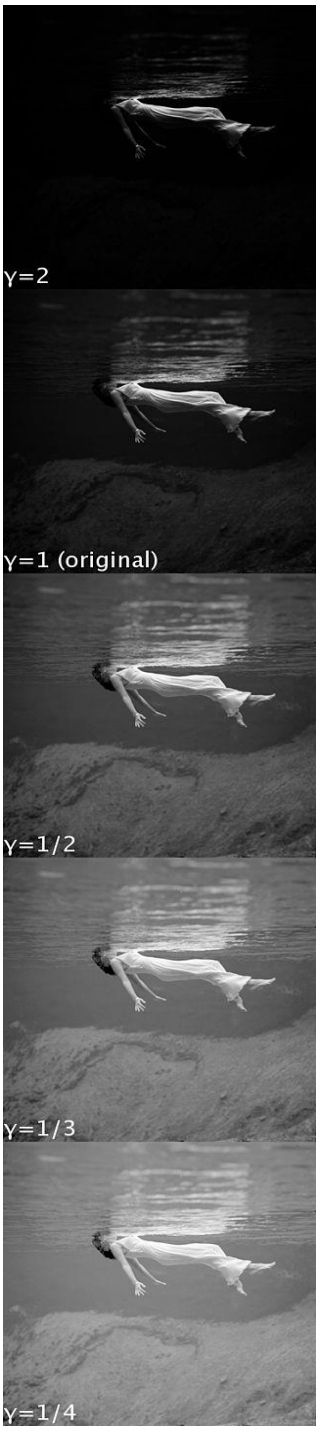


*Figura 4: Efecte de la correcció gamma d'una imatge*

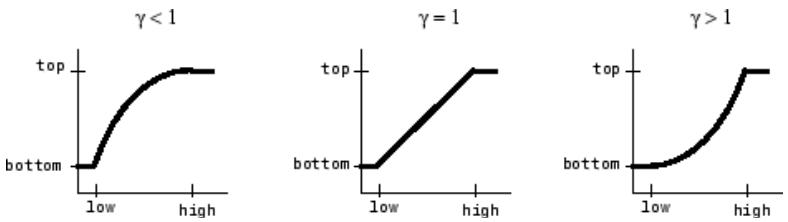


*Figura 5: Comportament de $V_{out}$ per diferents rangs de valors de $\gamma$*

Com nosaltres volem augmentar el contrast de les zones fosques de la imatge, fent que les zones més fosques de la nebulosa es facin més clares, haurem de fer servir un valor $\gamma < 1$ si decidim fer servir la correcció gamma. A la *Figura 6* es mostren, de dalt a baix, les imatges obtingudes amb valors $\gamma = 1/2, 1/3$ i $1/4$.

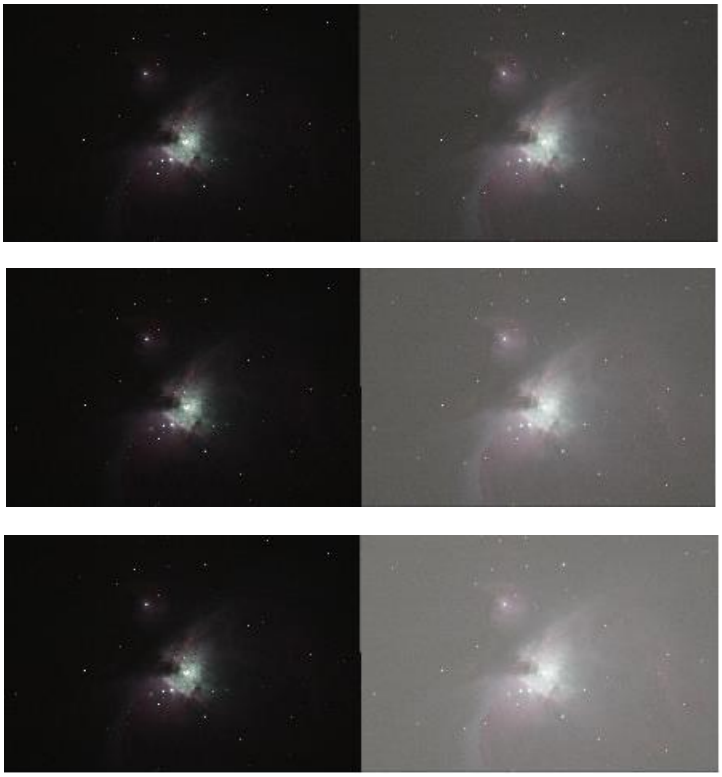

*Figura 6: Imatges obtingudes amb diferents valors de $V_{out}$ per diferents $\gamma$*

Notem que, de forma anàloga al que passava quan augmentàvem el valor de $k$ en l'anterior procediment, disminuir el valor de $\gamma$ provoca un contrast més brusc en la imatge resultant.

La crida que s'ha fet servir en MATLAB, per un determinat valor $lambda$, és:

```
17    Am = imadjust(Am,[],[],lambda);
```

## Replicant la solució de Shure

En aquesta última alternativa per augmentar el contrast de la imatge, he volgut intentar replicar el procediment seguit per Loren Shure, l'autor del *post*[1] original en el qual es basa l'exercici, ja que la idea que ella fa servir és diferent a les que hem proposat en aquest document, i m'ha semblat interessant de provar-la. Així que, en aquesta darrera secció explicarem els diferents passos que Shure segueix per il·luminar els detalls en les regions més fosques, a partir de la nostra imatge $Am$.

Primerament, Shure converteix la imatge RGB de partida a l'espai de colors HSL, obtenint així els components de to, saturació i lluminositat de la imatge, respectivament.

Recordem que el model de color HSL és una alternativa de representació del color al model RGB, que suposa una descripció més *intuïtiva*, en el sentit de la percepció del color per l'ull humà, dedicant dos canals (H i S) a la informació cromàtica del color, i un tercer component (L) a la lluminositat del color en qüestió (veure *Figura 7*).

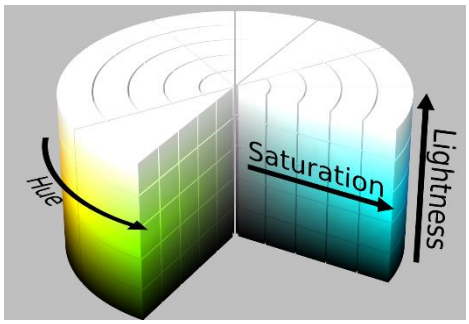

*Figura 7: Representació cilíndrica del model HSL*

Així, la idea de Shure és extreure els components HSL de la imatge de partida per ajustar la component L i, després, convertir la imatge modificada de nou a l'espai RGB. Amb tal propòsit es fan servir dues funcions: *convertToHSL* i *convertFromHSL*, que s'adjunten al final de document, per completitud.

Un cop extreta la component L, *aclarim* la lluminositat de la imatge, prenent el logaritme dels valors dels píxels en la component L.

```
17    [H,S,L] = convertToHSL(Am);
18    L2 = log(L);
```

Si mostrem l'histograma de la *nova* lluminositat de la imatge, veurem que l'eix $y$ de l'histograma es normalitza com la relació dels píxels amb una certa intensitat al nombre total de píxels de la imatge (*Figura 8*). Cal notar que la majoria dels píxels es corresponen a la regió del cel fosc.

```
19    histogram(L2,'Normalization','probability','EdgeColor','none');
20    xlabel('Adjusted lightness (a.u.)');
21    ylabel('Normalized pixel count');
```

A continuació, ajustem el nivell de negre de la imatge al valor d'intensitat en la regió del cel fosc. Podem re-escalar els valors de lluminositat al rang [0-1] i mostrar la imatge amb la lluminositat ajustada, per veure si hem escollit bé el llindar de foscor, o si hem de refinar aquest valor. El resultat que he obtingut pel valor de $blacklevel$ que he escollit es mostra a la *Figura 9*.

```
22    blacklevel = -2.98;
23    L2 = rescale(L2,"InputMin",blacklevel);
24    imshow(L2,[0 1])
```

[1] https://blogs.mathworks.com/loren/2020/11/06/astrophotography-with-matlab-imaging-the-orion-nebula/#33d27527-6983-4a05-b5b0-e612647b4e49

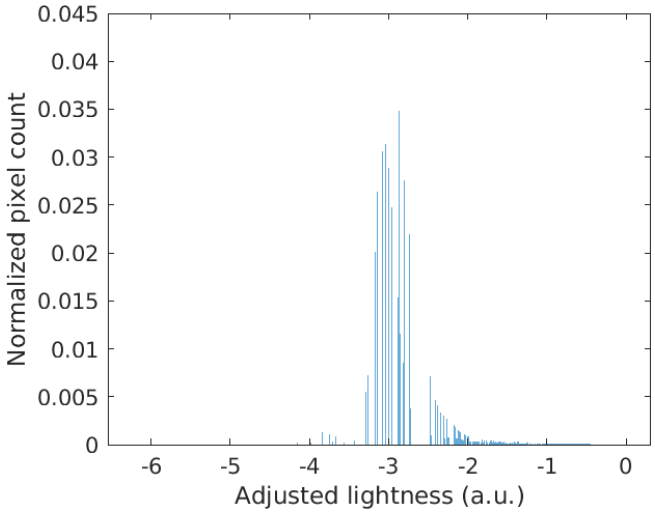


*Figura 8: Histograma del logaritme de la component L de la imatge*

*Figura 9: Imatge ajustada i re-escalada a l'interval [0-1]*

Finalment, cal convertir la imatge (havent ajustat la lluminositat) de nou a l'espai RGB. Si així ho desitgem, podem ajustar la saturació del color de la imatge final multiplicant la component S de la imatge original per un factor de saturació.

```
26    saturation = 0.9;
27    S2 = saturation*S;
28    Am = convertFromHSL(H,S2,L2);
```

A la següent figura es mostra la imatge obtinguda seguint aquest mètode amb el valor de re-ajust escollit que es mostra a la pàgina anterior (-2.98).

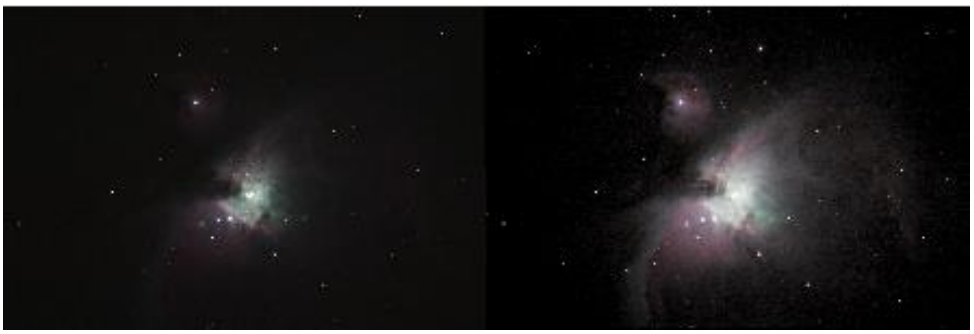

*Figura 10: Imatge obtinguda amb blacklevel = -2.98 i un factor de saturació de 0.90*

## Codi complert aplicant la solució de Shure

```
clear all;
close all;

A = double(imread('_MG_7735.JPG'))/255;
B = double(imread('_MG_7737.JPG')) /255;

DIF = abs(A-B); % imatge diferencia
maxim = max(DIF(:));
DIF = DIF/maxim; % dividim pel seu valor màxim
imshow(DIF);

Bd = imtranslate(B,[20, -20]);
DIF = abs(A-Bd);
maxim = max(DIF(:));
DIF = DIF/maxim;
imshow(DIF);

Am = (A+Bd)/2; % imatge millorada
[H,S,L] = convertToHSL(Am);
L2 = log(L);
histogram(L2,'Normalization','probability','EdgeColor','none');
xlabel('Adjusted lightness (a.u.)');
ylabel('Normalized pixel count');
blacklevel = -2.98;
L2 = rescale(L2,"InputMin",blacklevel);
imshow(L2,[0 1])

saturation = 0.9;
S2 = saturation*S;
Am = convertFromHSL(H,S2,L2);
montage ({A,Am});
```

```
function [H,SL,L] = convertToHSL(RGB)

HSV = rgb2hsv(RGB);

% Convert from HSV to HSL
% HL = HV = H;
H = HSV(:,:,1);
SV = HSV(:,:,2);
V = HSV(:,:,3);

L = V-(0.5.*V.*SV);

SL = (V-L) ./ min(L, 1-L);
SL(L==0 | L==1) = 0;
end
```

```
function RGB = convertFromHSL(H,SL,L)

% Convert from HSL to HSV
% HV = HL = H;
V = L+SL.*min(L, 1-L);
SV = 2*(1-L./V);
SV(V==0) = 0;

HSV = cat(3,H,SV,V);

RGB = hsv2rgb(HSV);

end
```