Team 03
Proposal Slide

Hello!

B04902083 莊翔旭
B04902048 蔡毓聰

A NEW COLOR CORRECTION METHOD FOR UNDERWATER IMAGING

G. Bianco [a*], **M. Muzzupappa** [b], **F. Bruno** [a], **R. Garcia** [b], **L. Neumann** [b,c]

[a] DIMEG, University of Calabria, 87036 Rende (CS), Italy
(gianfranco.bianco, muzzupappa, f.bruno)@unical.it

[b] Computer Vision and Robotics Group, University of Girona, 17001, Spain
rafael.garcia@udg.edu

[c] ICREA, Barcelona, Spain
lneumann@silver.udg.edu

1

Motivation

Waterproof camera?

防水相機興起

但拍出來的水下環境圖像大多都是藍藍綠綠的色調，且雜訊很多

有沒有可能可以透過演算法還原目標物在非水下環境的顏色？

水下作業拍攝

Extensions

- 水下考古文物還原真實色彩
- 幫助研究人員研究水下的生態狀況
- 套用其他影像處理，例如 3D 建模，物件辨識

Problem definition

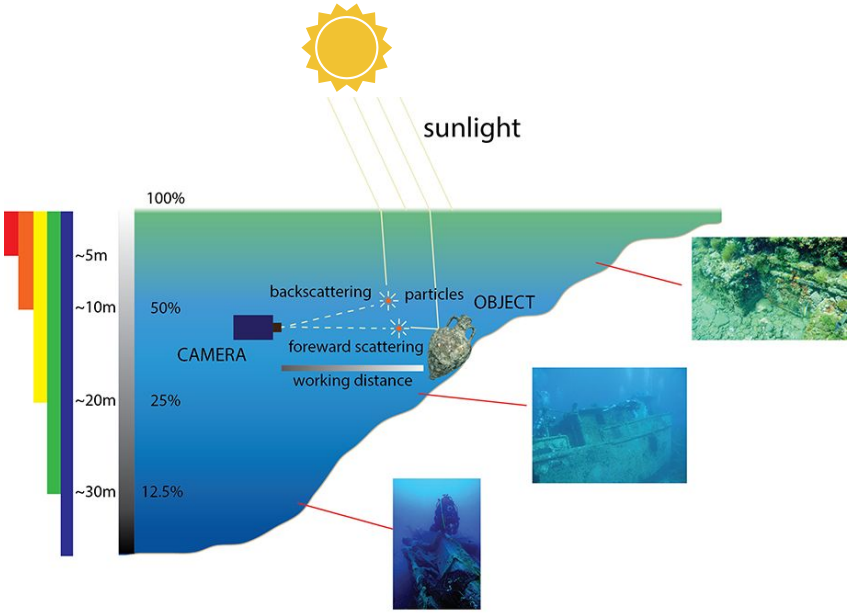
In minecraft: /fill 0 0 0 x y z air replace water

盡可能的
讓顏色還原正確

水下環境原圖像的目標物透過演算法
可以盡可能的還原成在非水下環境的
真實色彩。

盡可能的
讓顏色還原正確

光線穿過水會被分層吸收和散射，要還原顏色是很難的，也沒辦法確定結果是否是物體的真實色彩，所以是盡可能的接近。



Next Slide

3

Algorithm

Powered by ~~thanos infinity gauntlet~~ lαβ color space.

Assumptions

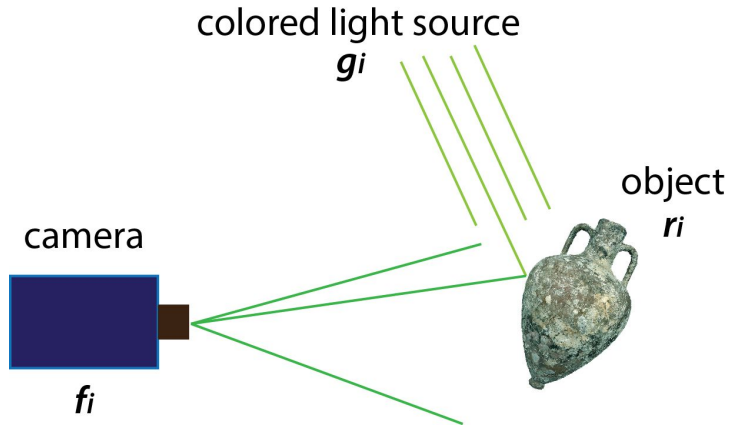
- Uniform light source
- Lambertian object surfaces
- Gray-world assumption

為了簡化複雜的水下的光線散射和折射變化，有以上的假設

Image Formation Model (by Gonzales and Woods, 1992)

$$f_i(m,n) = g_i(m,n) \times r_i(m,n)$$

$$g_i(m,n) = illuminant$$
$$r_i(m,n) = reflectance$$
$$i = \{R, G, B\}$$



Gray-world assumption

(by Buchsbaum, 1980)

假設對於有大量色彩變化的圖像，{R G B} 三個分量的平均值會趨於同一個 gray-level 。

換句話說，這篇論文的缺點是如果圖像沒有的色彩變化，假設就不成立，後續的處理效果就沒有這麼好，但大多數的自然風景圖像都會有大量的色彩變化。

最常用於白平衡的演算法中，消除光源的影響。

Gray-world Algorithm

(by Ebner, 2007)

$$f_i^*(m,n) = 1 \times r_i(m,n) = \frac{f_i(m,n)}{g_i(m,n)} \approx \frac{f_i(m,n)}{\overline{f_i}}$$

$$\text{where} \quad f_i^*(m,n) = \text{Output Image}$$
$$\overline{f_i} = mean\{f_i(m,n)\}$$

Image Formation Model

$g_i(m,n) = illuminant$

$r_i(m,n) = reflectance$

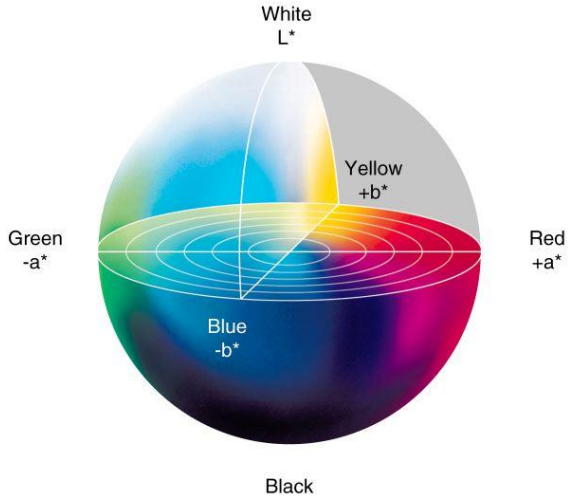
$i = \{R, G, B\}$

lαβ color space

與顯示設備無關且基於生理特征來描述人視覺感應的顏色系統。

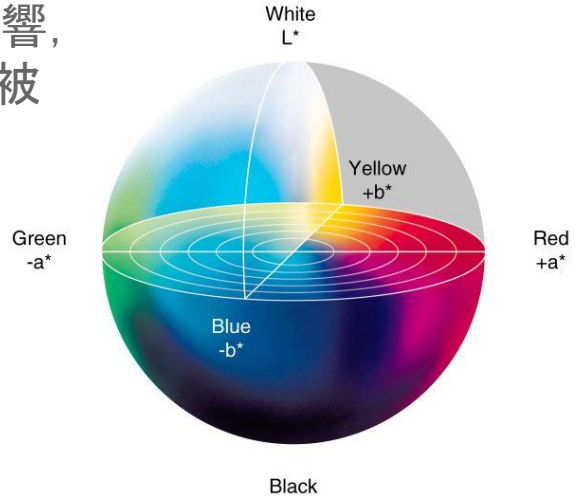
有三種 Channel：

- l 表示從純黑到純白，為像素的亮度；
- α 表示從紅色到綠色的範圍；
- β 表示從黃色到藍色的範圍；

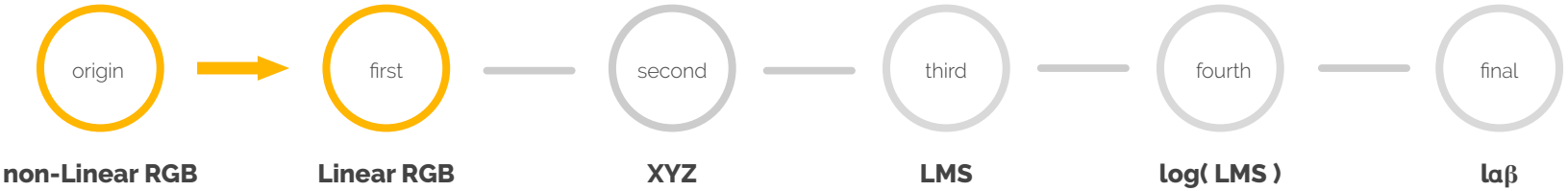


lαβ color space

- 正因為 lαβ 基於生理特征來描述人視覺感應，和人的視覺感知較為相近，且不會受顯示設備影響，很適合拿來做 color crrection，廣泛的被應用被各種色彩校準上。



RGB to lαβ (by Reinhard et al., 2001)



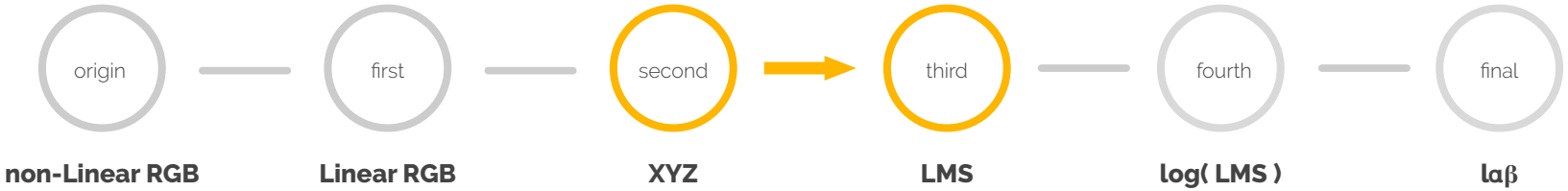
RGB image has to be corrected by the non linearity (**gamma correction**), to work with linear RGB coordinates.

RGB to lαβ (by Reinhard et al., 2001)

Linear RGB conversion in the XYZ tristimulus values.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

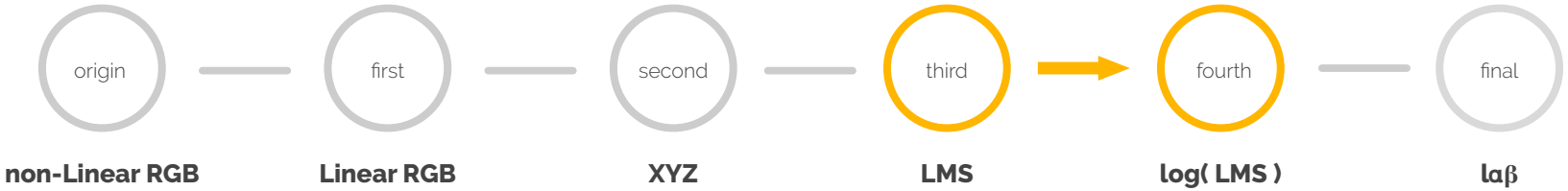
RGB to lαβ (by Reinhard et al., 2001)



From this device-independent XYZ space, we convert the image to LMS space.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

RGB to lαβ (by Reinhard et al., 2001)

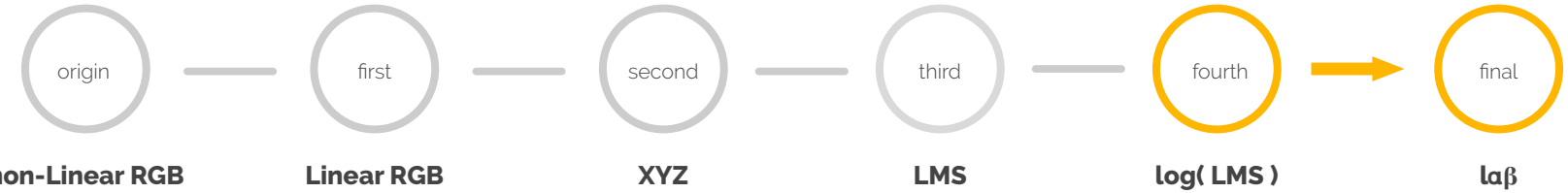


Transform the data in logarithmic space.

This step is a result of how our eyes detect lights non-linearly.

$$\begin{aligned} \boldsymbol{L} &= \log L \\ \boldsymbol{M} &= \log M \\ \boldsymbol{S} &= \log S \end{aligned}$$

RGB to lαβ (by Reinhard et al., 2001)

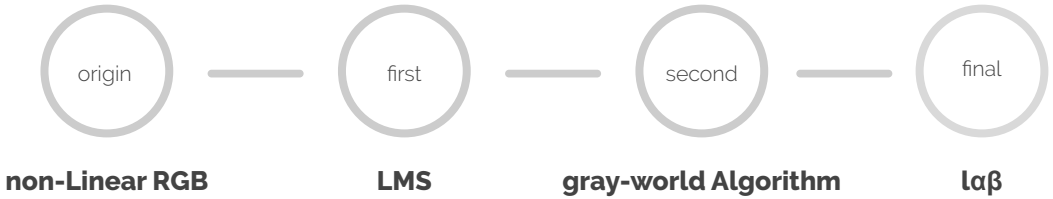


log(LMS) Transform to lαβ

$$\begin{aligned} \text{Achromatic} &\propto r+g+b \\ \text{Yellow–blue} &\propto r+g-b \\ \text{Red–green} &\propto r-g \end{aligned}$$

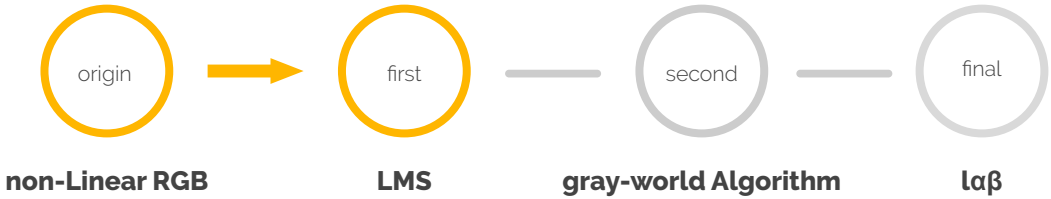
$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}$$

Color Correction



前面在定義需要用到的假設和其公式
現在要正式說明論文提出的 Color Correction 的演算法過程。

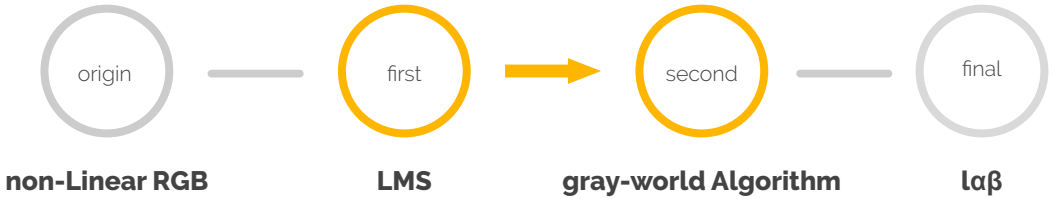
Color Correction



先依據 RGB to lαβ 的過程，將 origin 圖像轉成 LMS。

$$l_i(m,n) = T_{rgb \to xyz} \times T_{xyz \to lms} \times f_i(m,n)$$

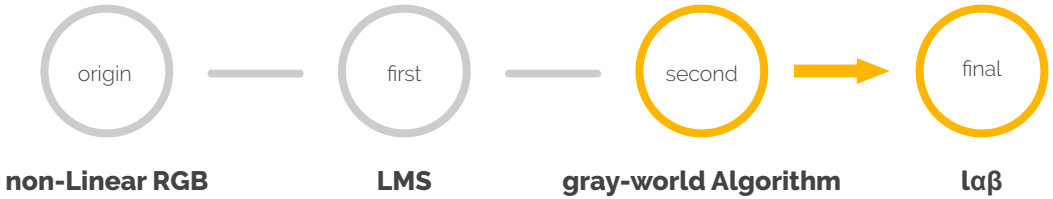
Color Correction



在 LMS space 套用 Gray World Algorithm

$$l_i^*(m,n) \approx \frac{l_i(m,n)}{\bar{l}_i}$$

Color Correction



將做完 gray-world Algorithm 的 LMS 轉成 lαβ，得到以下公式：

$$\log_{10} l_i^*(m,n) \approx \log_{10} \frac{l_i(m,n)}{\overline{l_i}} = \log_{10} l_i(m,n) - \log_{10} \overline{l_i}$$

$$T_{LMS\to l\alpha\beta} \times \log_{10} l_i^*(m,n) \approx T_{LMS\to l\alpha\beta} \times \log_{10} l_i(m,n) - T_{LMS\to l\alpha\beta} \times \log_{10} \overline{l_i}$$

$$l^*_{l\alpha\beta,i}(m,n) \approx l_{l\alpha\beta,i}(m,n) - \overline{l_{l\alpha\beta,i}}$$

Color Correction

final

lαβ

將得到的公式對 { α, β } channel 做處理

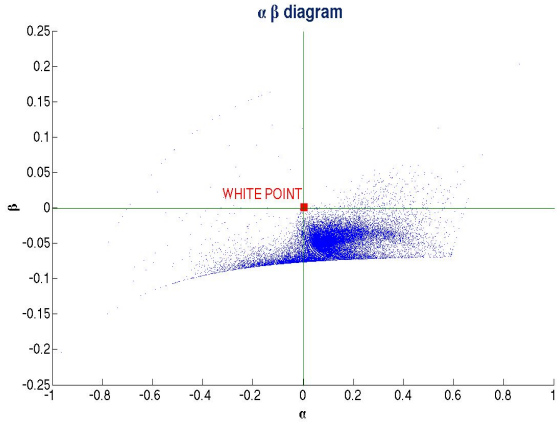
$$l^*_{l\alpha\beta,i}(m,n) \approx l_{l\alpha\beta,i}(m,n) - \overline{l_{l\alpha\beta,i}}$$

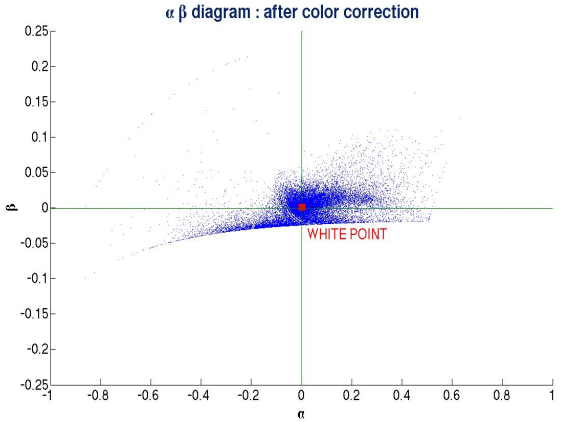
Color Correction

final

lαβ

值得一提的是，原先的 gray-world algorithm 會把 RGB space 平均移到 {1, 1, 1}, 而在 lαβ space 底下則會跑到 {0, 0, 0}



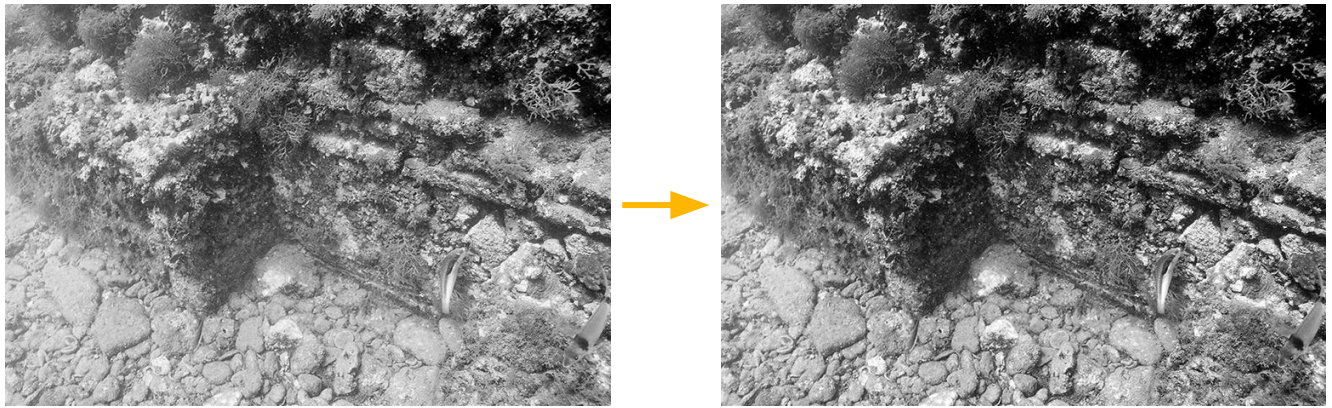


Color Correction

final

lαβ

{ l } channel 也可以不做 gray-world Algorithm，做了可以增加 illumination 的對比度，另外也可以做 histogram stretching 達到差不多的效果。



4

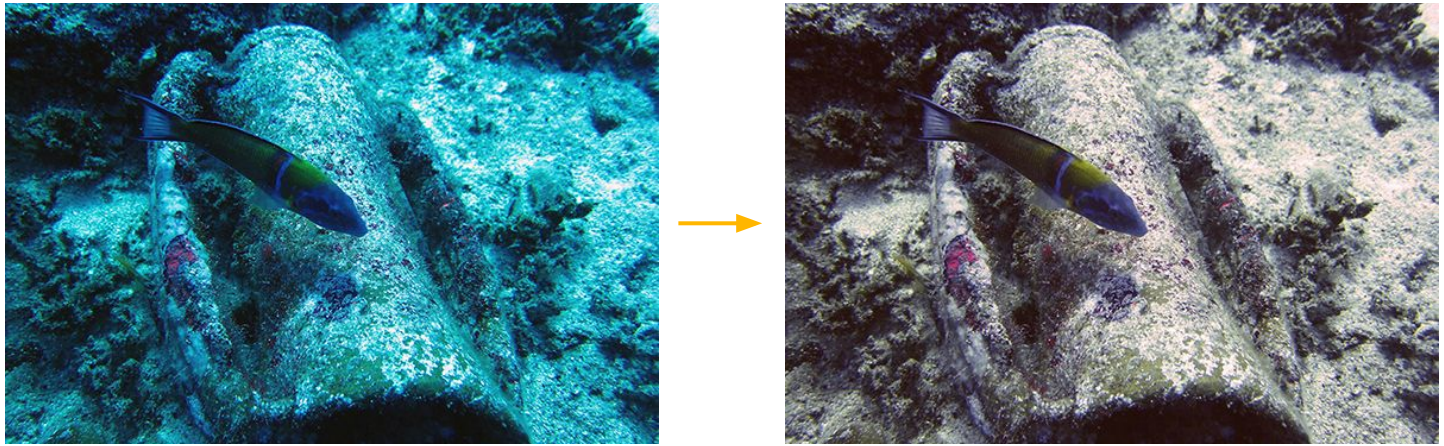
Expected results

Part of a brick wall

in which a greenish component is present.

Amphora (Kas, Turkey)

in which a bluish component is dominant.



Real-time implement

因為在應用上大多都需要快速預覽結果，希望越快越好。本篇論文的方法在實作上時間複雜度為 big-O(n) by n = Width*Height

作者在 Mac 上使用 matlab 實作，591x892 的照片平均 0.7 秒
Apple Mac SPEC：

- Processor 2.7 GHz Core i7
- 16 GB 1600 MHz DDR3 Memory
- NVIDIA GeForce GT 650M 1024 MB

5

Reference

Image sources

1. P05. background:
   - http://news.algaeworld.org/2016/05/underwater-forests-restored-off-sydneys-iconic-beaches/
2. P10. Drake meme:
   - https://youtu.be/uxpDa-c-4Mc?t=78
3. P16. Lab Chart
   - https://genialfoto.com/bild/lab-color-chart-e1.html

Thanks!

Any questions?