Apilados de imagen con R

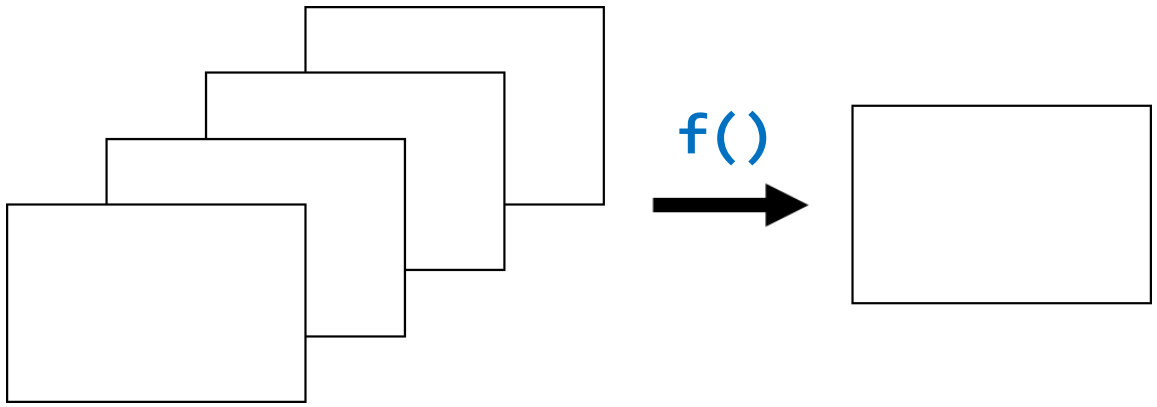
Guillermo Luijk
overfitting.net
Madrid, junio 2024

Apilados de imagen con R

1. INTRODUCCIÓN
2. ARCHIVOS **RAW** DE IMAGEN
3. DESCOMPOSICIÓN **LINEAL** DE LUZ
4. APILADO POR **MEDIA**
5. APILADO POR **MEDIANA**
6. APILADO POR **'ANTIMEDIANA'**
7. APILADO POR **MÁXIMO**
8. APILADO POR **MÍNIMO**
9. APILADO **HDR**

1. INTRODUCCIÓN

- **Apilado** = combinación de una secuencia de imágenes para obtener una imagen final de características nuevas o mejoradas
- Explotaremos estadísticamente varias **funciones de agregación**
- Implementación matricial en **R base** portable a otros lenguajes



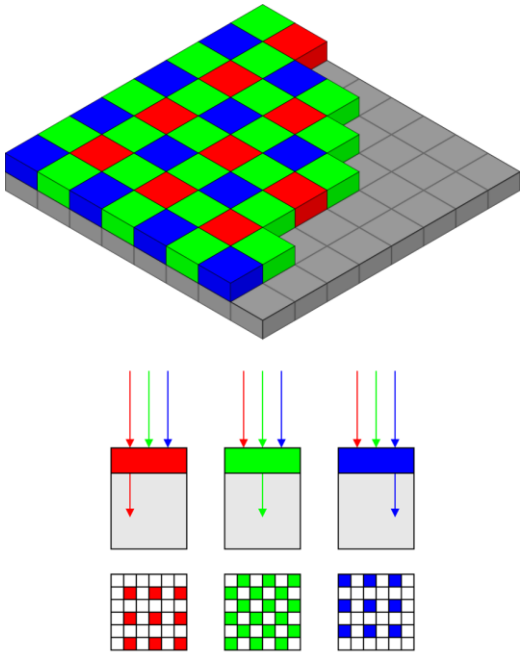
1. INTRODUCCIÓN

Función de apilado	Utilidad	Debilidades	Coste CPU
Media `mean()`	- Reducir **ruido**/ganar rango dinámico - Simular **larga exposición**	- Fantasmas con sujetos móviles	Bajo
Mediana `median()`	- Reducir ruido/ganar rango dinámico - Eliminar **sujetos móviles**		Alto
Antimediana `antimedian()`	- Replicar **sujetos móviles**	- Artefactos en solapes	Alto
Máximo `max()`	- Acumular luz (**star trails**, light painting, fotografía nocturna,...)	- Artefactos con sujetos móviles	Medio
Mínimo `min()`	- Preservar **sombras**	- Artefactos con sujetos móviles	Medio
HDR	- Reducir ruido/ganar **rango dinámico**	- Artefactos con sujetos móviles	Medio

`apply()`

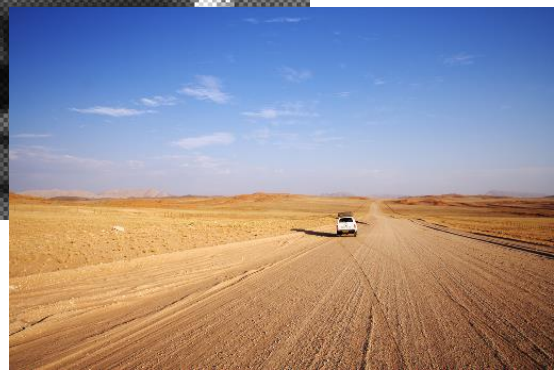
2. ARCHIVOS RAW DE IMAGEN

sensor Bayer



RAW

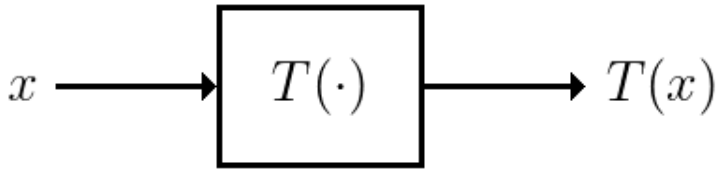
JPEG



flujo de revelado RAW

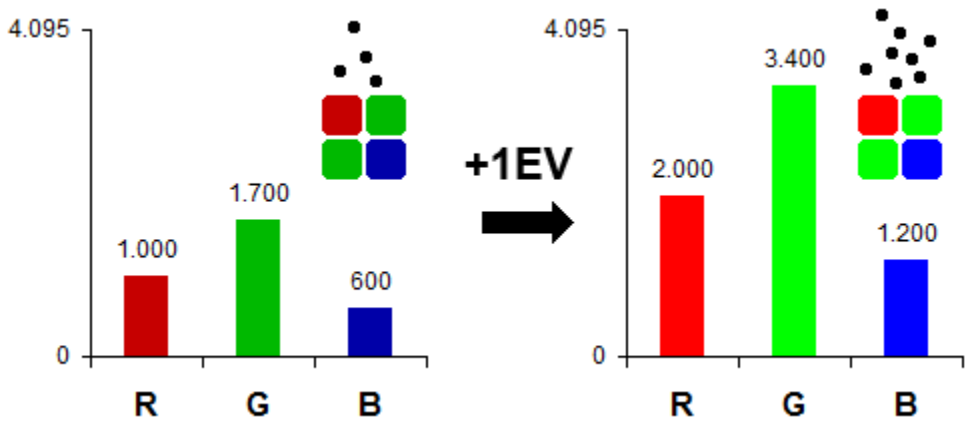
2. ARCHIVOS **RAW** DE IMAGEN

Sistema Lineal = aquél que mantiene suma y producto por escalar



$$T(\alpha\, x + \beta\, y) = \alpha\, T(x) + \beta\, T(y)$$

Un **sensor de imagen** es un “contador de fotones” lineal



valores RAW

2. ARCHIVOS RAW DE IMAGEN

flujo de implementación de los apilados en RAW

RAW

Adobe DNG Converter

DNG

DCRAW

```
dcraw -v -D -t 0 -4 -T *.dng
dcraw -v -d -t 0 -r 1 1 1 1 -4 -T *.dng
```

TIFF
(Bayer)

Photoshop, ImageMagick,
R, Python,...

Output TIFF
(Bayer)

exiftool
dng_validate

revelador RAW

DNG
(16 bits)

3. DESCOMPOSICIÓN LINEAL DE LUZ

Objetivo: aislar fuentes de iluminación con **operaciones lineales**

$$RAW(Amb + Art) = RAW(Amb) + RAW(Art)$$

$$\downarrow$$

$$RAW(Art) = RAW(Amb + Art) - RAW(Amb)$$

3. DESCOMPOSICIÓN LINEAL DE LUZ

estimación de
iluminación artificial:

$$RAW(Art) = RAW(Amb + Art) - RAW(Amb)$$

validación del resultado:
captura en presencia solo
de iluminación artificial
(persianas bajadas)

3. DESCOMPOSICIÓN LINEAL DE LUZ

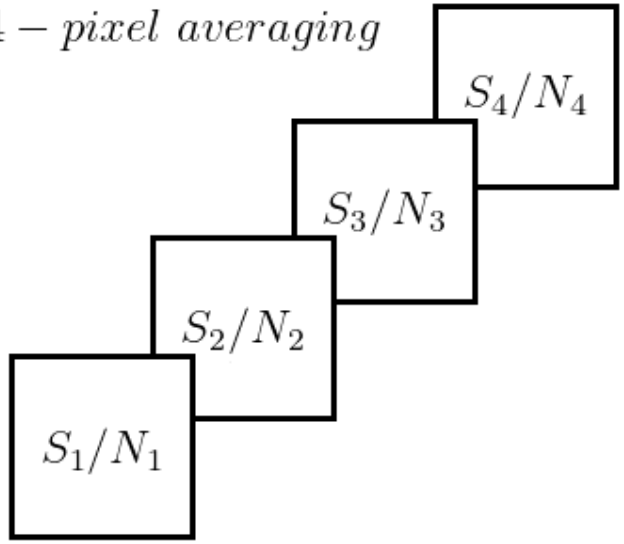
Código R:

```
# PARAMETERS
N=2  # number of RAW files to process
NAME="raw"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# RAW files are named:
#   raw1.tiff: (Ambiente + Artificial)
#   raw2.tiff: (Ambiente)

# READ RAW DATA
img=list()
for (i in 1:N) {
    img[[i]]=readTIFF(paste0(NAME, i, ".tiff"),
                      native=FALSE, convert=FALSE)
}

# LINEAR SUBTRACTION
imag=img[[1]]-img[[2]]  # (Ambiente + Artificial) - (Ambiente)

# BUILD OUTPUT DNG
writeTIFF(imag/max(imag), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

4. APILADO POR MEDIA

Objetivo: reducir **ruido**/ganar rango dinámico o emular tomas de **larga exposición**

`mean()`



$$S_T = S_1 + S_2 + S_3 + S_4 = 4S$$

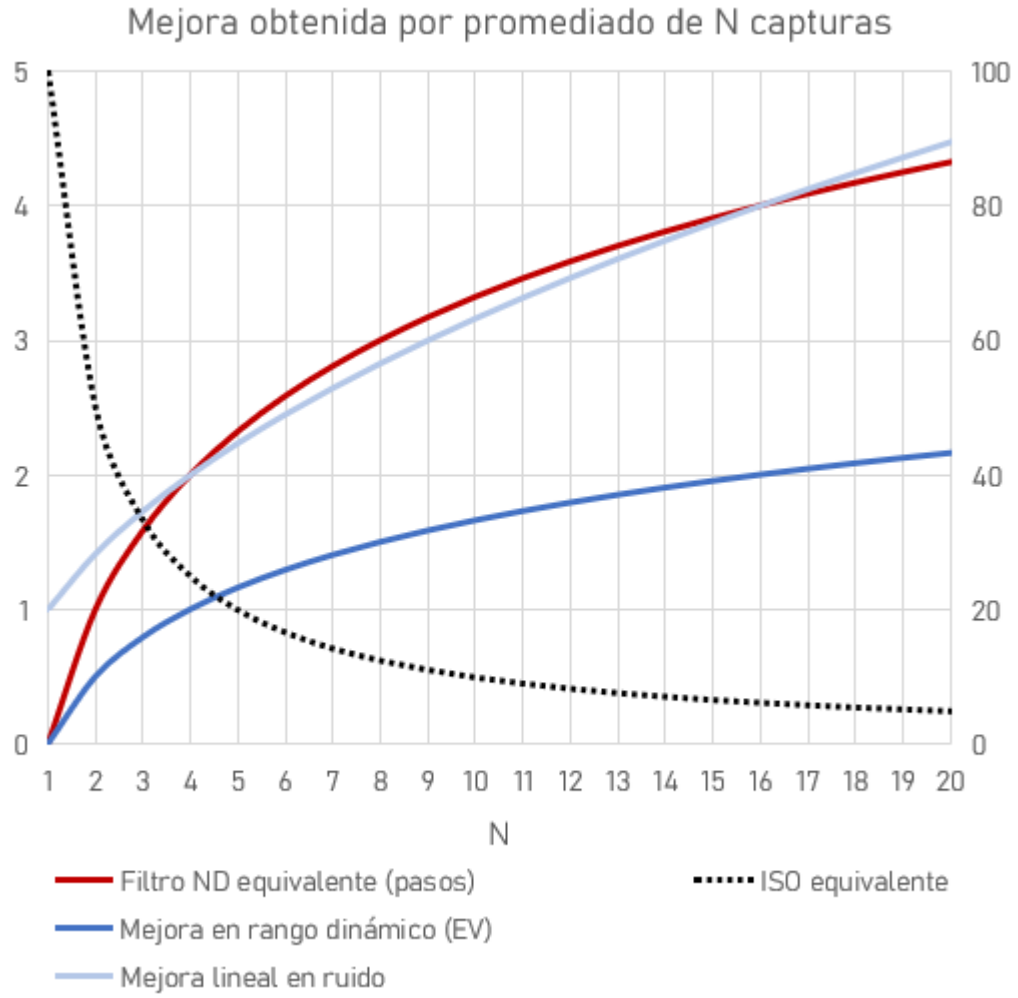
$$N_T^2 = N_1^2 + N_2^2 + N_3^2 + N_4^2 = 4N^2$$
$$\rightarrow N_T = 2N$$

$$\left(\frac{S}{N}\right)_T = \frac{S_T}{N_T} = \frac{4S}{2N} = 2\left(\frac{S}{N}\right)$$

4. APILADO POR MEDIA

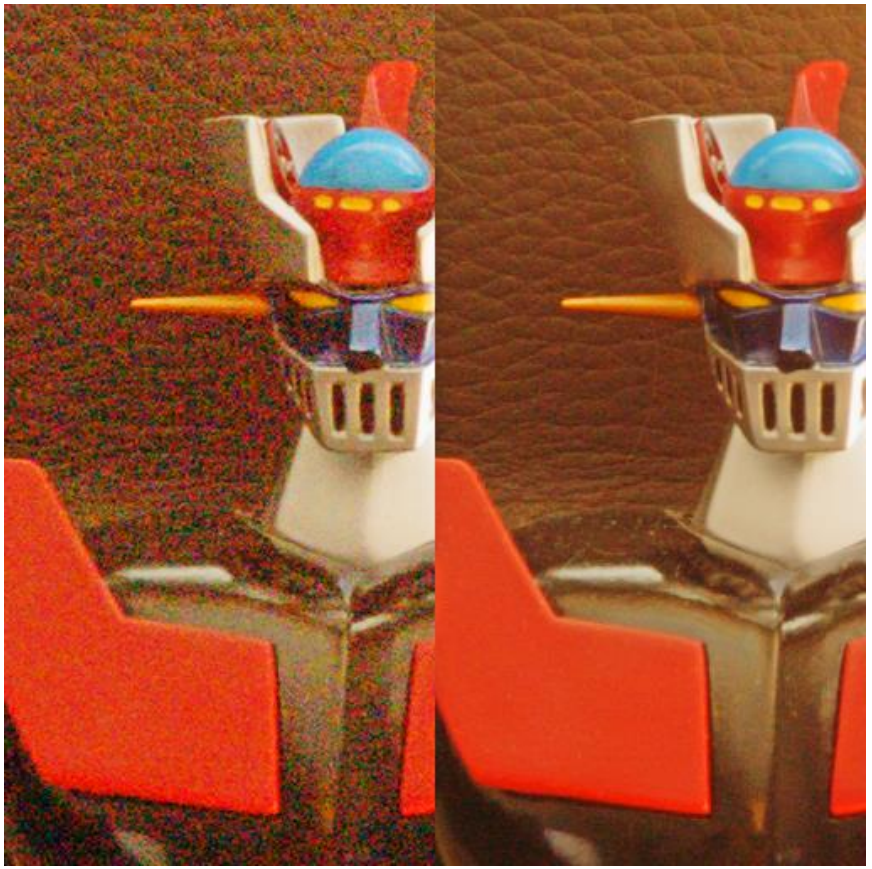
Objetivo: reducir **ruido**/ganar rango dinámico o emular tomas de **larga exposición**

`mean()`



4. APILADO POR MEDIA

Uso 1: reducir **ruido**/ganar rango dinámico (~ISO muy bajo)



– Capturas: **16 tomas** a ISO100

– Equivalencia ISO 100/16 = **ISO6**

– Mejora en rango dinámico: $\log_4(16)$ = **2EV**

– Mejora lineal en ruido: $16^{1/2}$ = **4**

4. APILADO POR MEDIA

Uso 2: emular tomas de **larga exposición** (~filtro ND)

- Capturas: **14 tomas** a ISO100 de 30s
- Tiempo de exposición equiv.: 0,5 · 14 = **7min**
- Equivalencia ISO 100/14 = **ISO7**
- Equivalencia filtro ND de $\log_2(16)$ = **3,81 pasos**
- Mejora en rango dinámico: $\log_4(14)$ = **1,90EV**
- Mejora lineal en ruido: $14^{1/2}$ = **3,74**

4. APILADO POR MEDIA

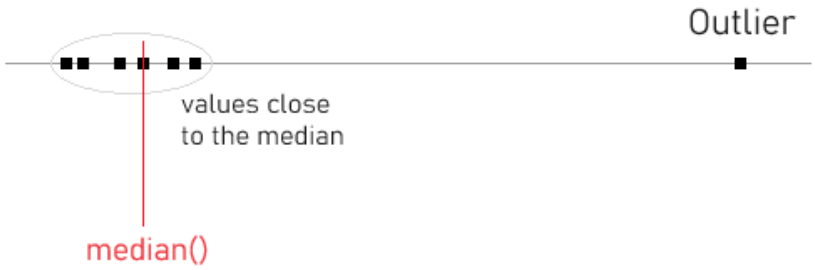
Código R:

```
# PARAMETERS
N=16  # number of RAW files to merge
NAME="_DSC38"  # input RAW filenames
INIT=18  # first file
OUTNAME="bayer"  # output RAW composite filename

# READ RAW DATA
img=0
for (i in 1:N) {
    name=paste0(NAME, i+INIT-1, ".tiff")
    img=img+readTIFF(name, native=FALSE, convert=FALSE)
}

# MEAN AVERAGING
img=img/N

# BUILD OUTPUT DNG
writeTIFF(img/max(img), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

5. APILADO POR MEDIANA

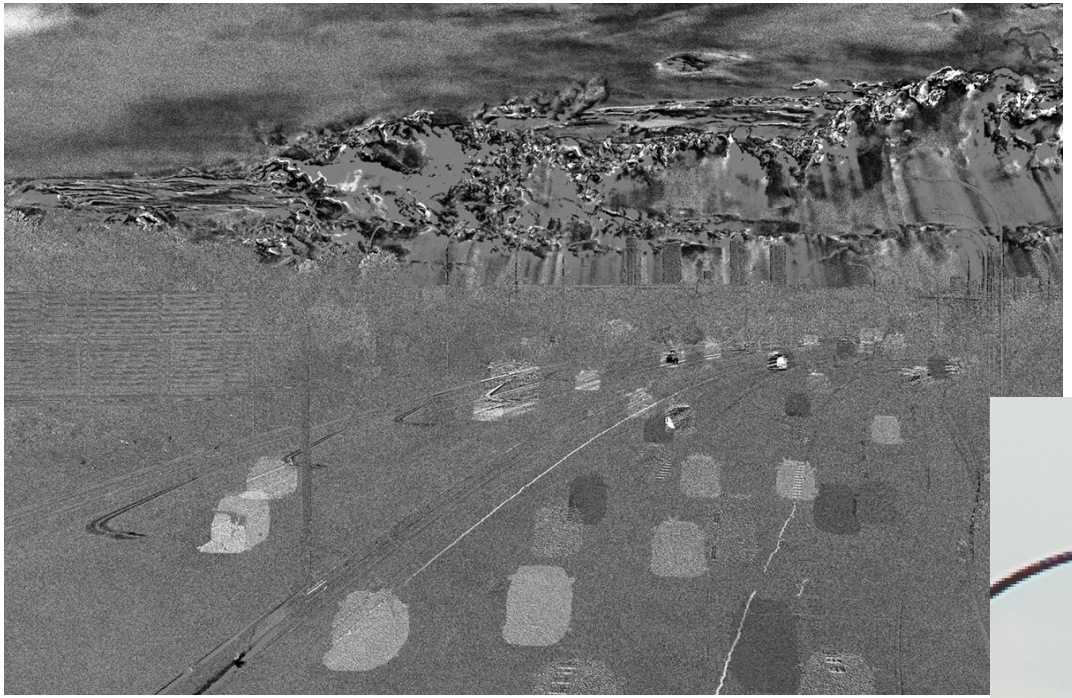
Objetivo: eliminar sujetos **en movimiento**

5 tomas consecutivas

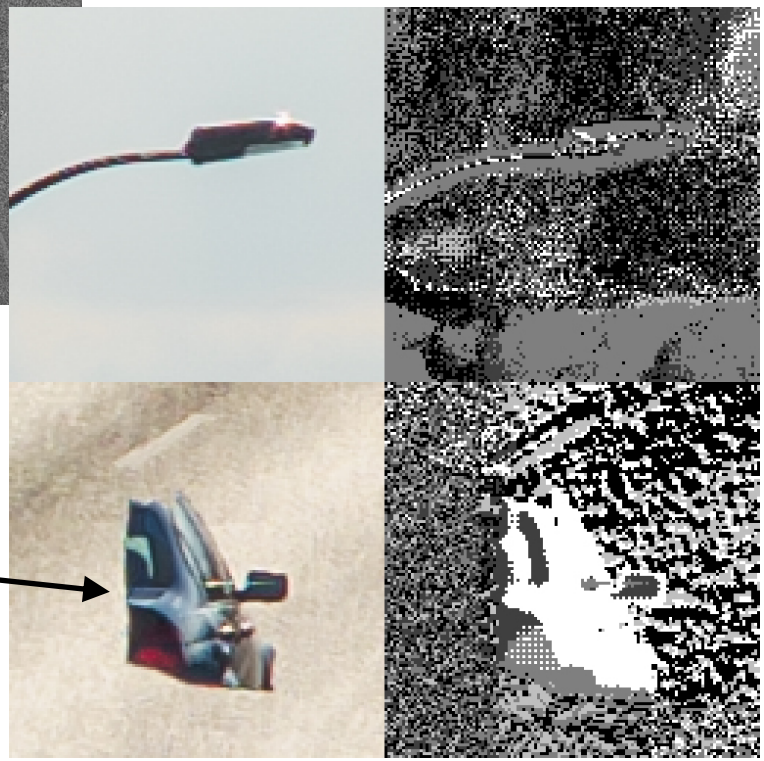


`median()`

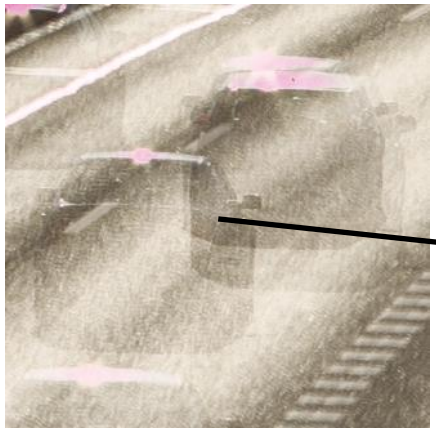
5. APILADO POR MEDIANA



mapa de fusión



residuos de
la mediana



5. APILADO POR MEDIANA

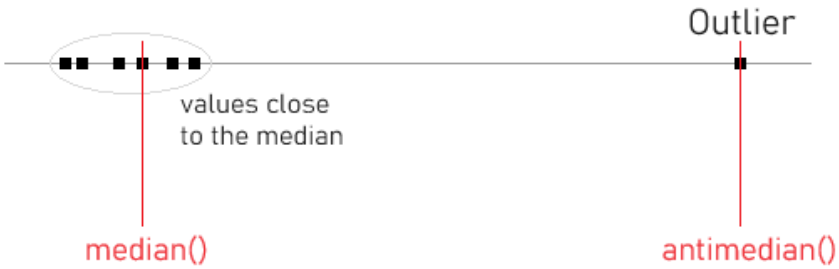
Código R:

```
# Improve the performance of the R median function with this C++ code:
library(Rcpp)
cppFunction('
            double cpp_med2(Rcpp::NumericVector xx) {
            Rcpp::NumericVector x = Rcpp::clone(xx);
            std::size_t n = x.size() / 2;
            std::nth_element(x.begin(), x.begin() + n, x.end());
            if (x.size() % 2) return x[n];
            return (x[n-1] + x[n]) / 2.;
            }
            ')

# PARAMETERS
N=5  # number of RAW files to merge
NAME="raw"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# READ RAW DATA
img=readTIFF(paste0(NAME, 1, ".tiff"), native=FALSE, convert=FALSE)
img=array(0, c(nrow(img), ncol(img), N))
for (i in 1:N) {
    img[,,i]=readTIFF(paste0(NAME, i, ".tiff"), native=FALSE, convert=FALSE)
}

# MEDIAN AVERAGING
# median: Time difference of 14.6095 mins
imag=apply(img, c(1,2), median)  # c(1,2) means 1st and 2nd dimensions

# cpp_med2: Time difference of 1.206512 mins (~12 times faster)
imag=apply(img, c(1,2), cpp_med2)  # c(1,2) means 1st and 2nd dimensions

# BUILD OUTPUT DNG
writeTIFF(imag/max(imag), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

6. APILADO POR ‘ANTIMEDIANA’

Objetivo: preservar sujetos **en movimiento**

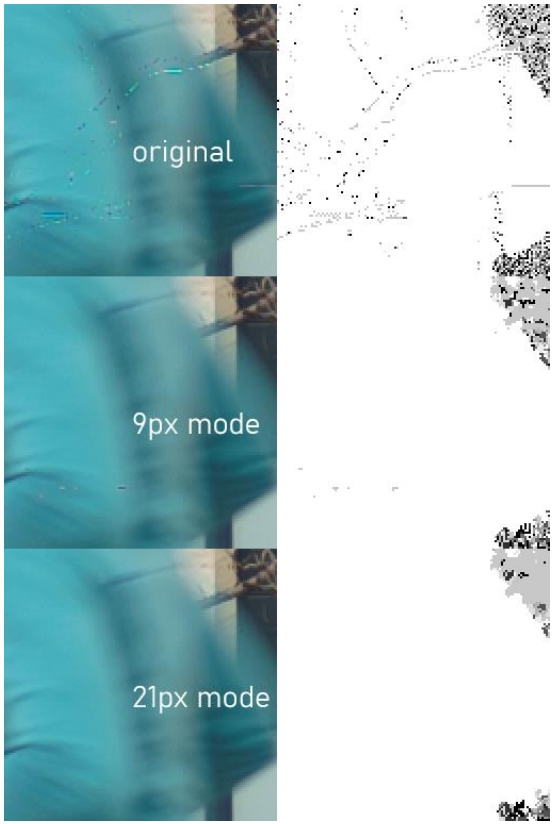
11 tomas en ráfaga



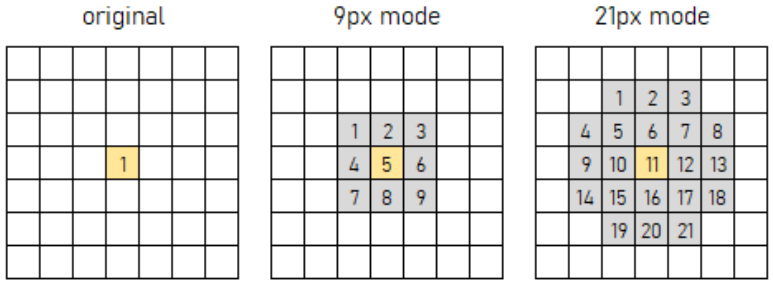
`antimedian()`

6. APILADO POR 'ANTIMEDIANA'

mapa de fusión



eliminación de artefactos usando la moda



6. APILADO POR 'ANTIMEDIANA'

Código R:

```
# Custom 'antimedian' function
antimedian = function(x) {
    absdev = abs(x - cpp_med2(x))  # absdev = abs(x - median(x))
    x[absdev == max(absdev)][1]  # [1] to return single value
}

# Statistical mode function
statmode = function(x) {
    ux=unique(x)
    ux[which.max(tabulate(match(x, ux)))]
}

# PARAMETERS
N=11  # number of RAW files to merge
NAME="raw"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# READ RAW DATA
img=readTIFF(paste0(NAME, 1, ".tiff"), native=FALSE, convert=FALSE)
img=array(0, c(nrow(img), ncol(img), N))
for (i in 1:N) {
    img[,,i]=readTIFF(paste0(NAME, i, ".tiff"),
                      native=FALSE, convert=FALSE, as.is=TRUE)
}

# 'ANTIMEDIAN' AVERAGING
imag=apply(img, c(1,2), antimedian)

# BUILD OUTPUT DNG
writeTIFF(imag/max(imag), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

7. APILADO POR MÁXIMO

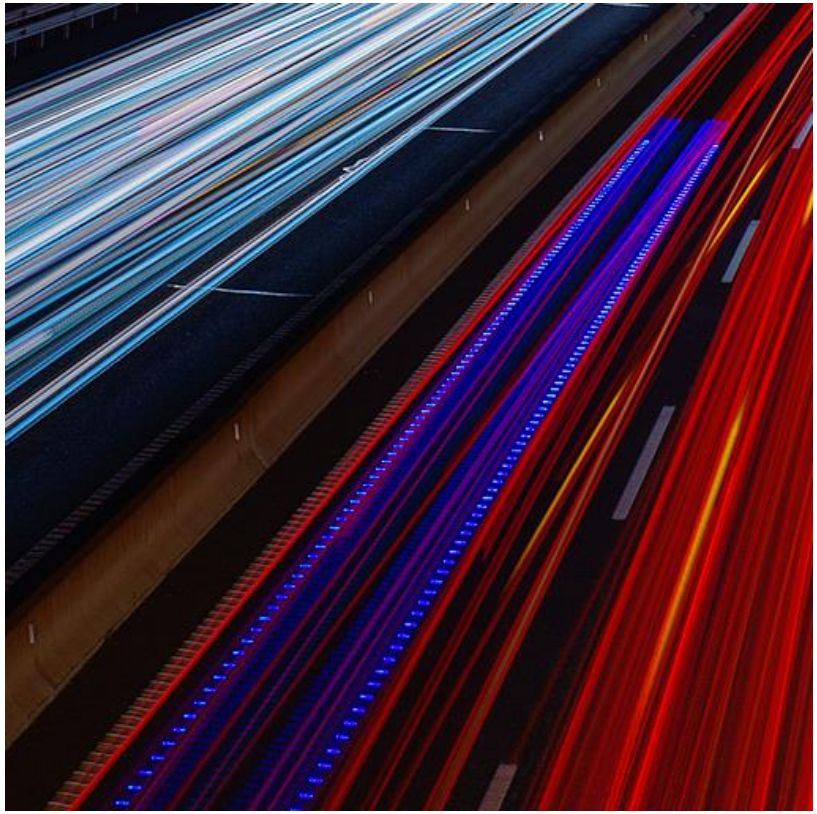
Objetivo: acumular **luz** (fotografía nocturna, light painting,…)

12 tomas de 20”

`max()`

7. APILADO POR **MÁXIMO**

artefactos debidos a
sujetos en movimiento



luces pulsantes (LED)
generan discontinuidades

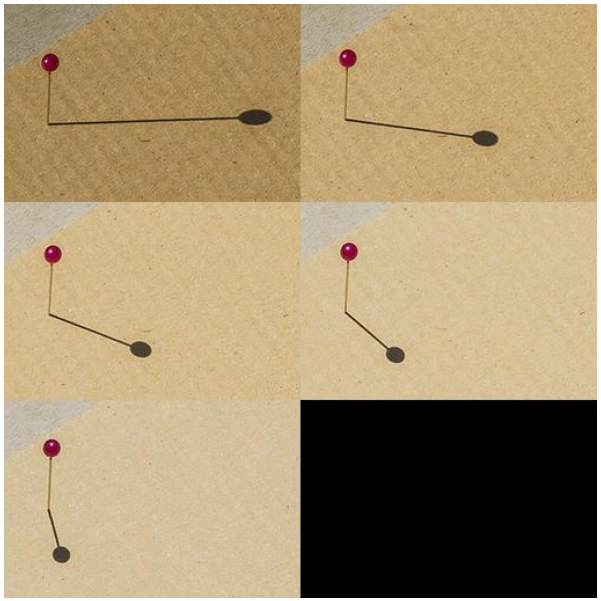
7. APILADO POR MÁXIMO

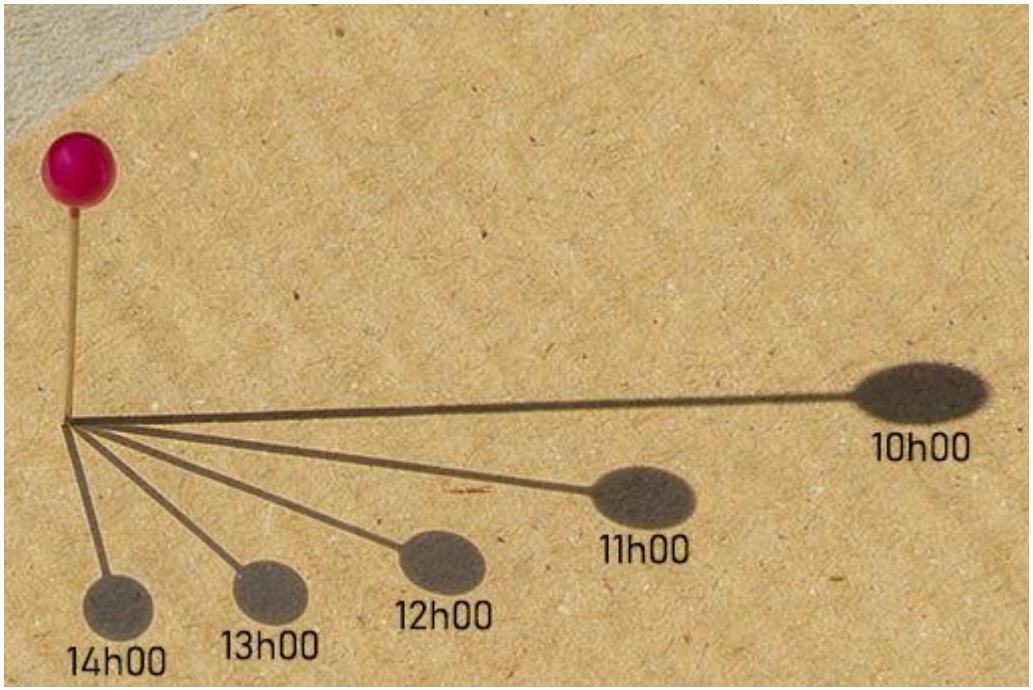
Código R:

```
# PARAMETERS
N=12  # number of RAW files to merge
NAME="raw"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# READ RAW DATA AND CALCULATE MAX
imag=readTIFF(paste0(NAME, 1, ".tiff"), native=FALSE, convert=FALSE)
for (i in 2:N) {
    img=readTIFF(paste0(NAME, i, ".tiff"), native=FALSE, convert=FALSE)
    indices=which(img>imag)  # keep any higher value
    imag[indices]=img[indices]
}

# BUILD OUTPUT DNG
writeTIFF(imag/max(imag), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

8. APILADO POR MÍNIMO

Objetivo: preservar **sombras**





`min()`

5 tomas separadas 1h

8. APILADO POR MÍNIMO

Código R:

```
# PARAMETERS
N=5  # number of RAW files to merge
NAME="DSC0551"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# READ RAW DATA AND CALCULATE MIN
imag=readTIFF(paste0(NAME, 1, ".tiff"), native=FALSE, convert=FALSE)
for (i in 2:N) {
    img=readTIFF(paste0(NAME, i, ".tiff"), native=FALSE, convert=FALSE)
    indices=which(img<imag)  # keep any lower value
    imag[indices]=img[indices]
}

# BUILD OUTPUT DNG
writeTIFF(imag/max(imag), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

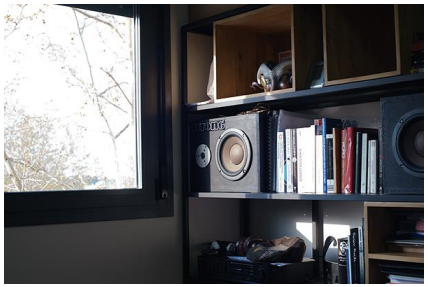
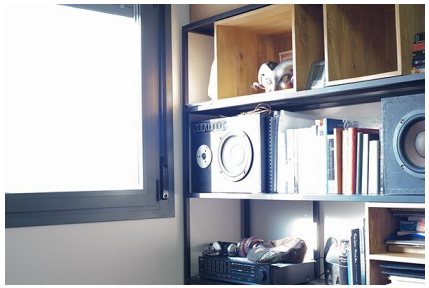
9. APILADO HDR

Objetivo: reducir **ruido** y aumentar el **rango dinámico**

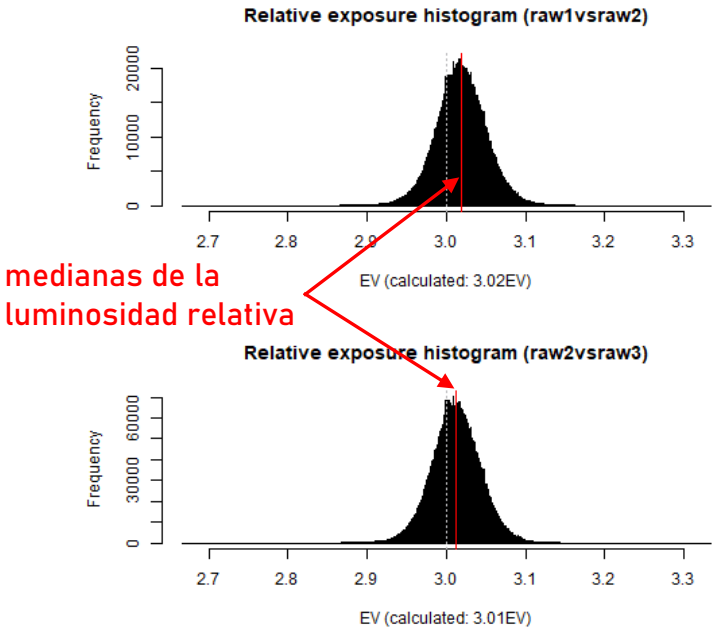
HDR (High Dynamic Range) = técnica de captura en la cual se realizan tomas de diferente exposición:

- Tomas menos luminosas → altas luces
- Tomas más luminosas → sombras

9. APILADO HDR

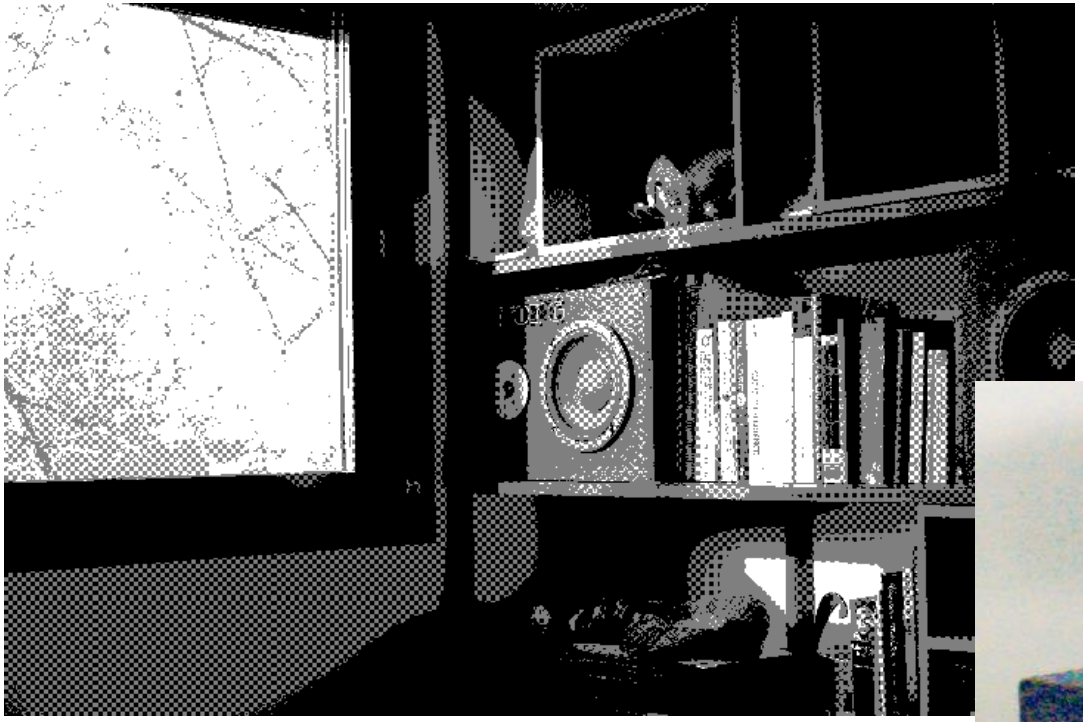




3 tomas separadas 3EV {0, +3EV, +6EV}

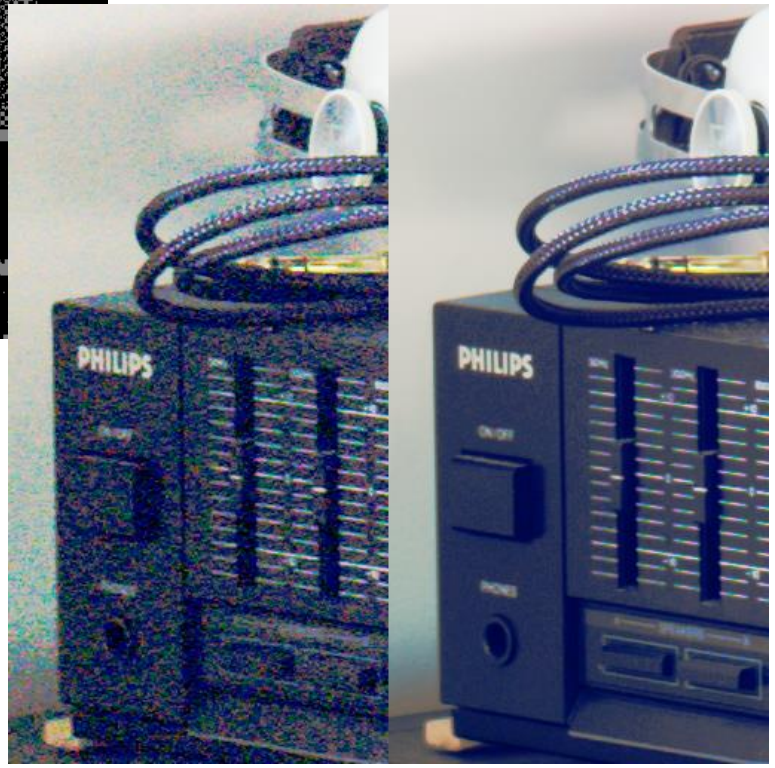


medianas de la luminosidad relativa

9. APILADO HDR



mapa de fusión



comparación de ruido

9. APILADO HDR

Código R:

```
# PARAMETERS
N=3  # number of RAW files to merge
NAME="raw"  # input RAW filenames
OUTNAME="bayer"  # output RAW composite filename

# Linear valid exposure range
MIN=2^(-5)  # from -5EV... (acceptable noise)
MAX=2^(-1/2)  # ...up to -1/2EV (avoid non-linearity)

# READ RAW DATA
img=list()
for (i in 1:N) {
    img[[i]]=readTIFF(paste0(NAME, i, ".tiff"), native=FALSE, convert=FALSE)
}

# RELATIVE EXPOSURE CALCULATIONS
f=array(-1, N-1)
for (i in 1:(N-1)) {
    indices=which(img[[i]]  >=MIN & img[[i]]  <=MAX &
                  img[[i+1]]>=MIN & img[[i+1]]<=MAX)
    exprel=img[[i+1]][indices]/img[[i]][indices]
    f[i]=median(exprel)  # linear exposure correction factor
}

# BUILD HDR COMPOSITE
hdr=img[[1]]  # start with lowest exposure data
for (i in 2:N) {
    indices=which(img[[i]]<=MAX)  # non-clipped highest exposure
    hdr[indices]=img[[i]][indices]/cumprod(f)[i-1]  # overwrite+exp correction
}

writeTIFF((hdr/max(hdr)), paste0(OUTNAME,".tif"), bits.per.sample=16,
          compression="none")
```

gracias

overfitting.net