

Μετατροπή RAW σε RGB

Οδυσσέας Σοφικίτης, 10130
sodyssea@ece.auth.gr

1 Ανάγνωση Εικόνας

Σε μεγάλο βαθμό η ανάγνωση του *.dng* αρχείου γίνεται από τον κώδικα που παρουσιάζεται στην εξόφληση. Αφού εξάγουμε τα διάφορα χαρακτηριστικά της φωτογραφίας (*rawim*, *wbcoeffs* κτλ), κρατάμε μόνο το χρήσιμο κομμάτι του πίνακα με τις μετρήσεις των αισθητήρων, χρησιμοποιώντας τις διαστάσεις που πήραμε από το *meta_info* και κανονικοποιούμε τις τιμές αυτές αφού διορθωθούν με το *blacklevel* (Σχήμα 1.1).

Η μόνη διαφορά σε σχέση με την κλασική ανάγνωση μιας *raw* φωτογραφίας είναι πως έχει προστεθεί στο τέλος ένας έλεγχος (Σχήμα 1.2) ώστε να διατηρείτε η κάθε διάσταση της φωτογραφίας ζυγός αριθμός. Φυσικά, αυτό ισχύει στην πλειονότητα των περιπτώσεων, αλλά είναι σημαντικό να επιβεβαιώνεται αφού στο *demosaicing* θεωρείται ότι το *Bayer pattern* διατηρείται αναλογικό στις άκρες.

```
% % correct info
% keep true size
rawim = double(rawim(y_origin : y_origin + height - 1,
                      x_origin : x_origin + width - 1));

% subtract possible black offset due to noise
rawim = rawim - blacklevel;
% normalize 0-1
rawim = rawim / (whitelevel - blacklevel);
% correct out of bounds| values
rawim = max(0, min(1, rawim));
```

Σχήμα 1.1: Διόρθωση διαστάσεων και κανονικοποίηση

```
% % keep size as an even number
if mod(2, size(rawim, 1)) == 1
    rawim(end, :) = [];
end
if mod(2, size(rawim, 2)) == 1
    rawim(:, end) = [];
end
```

Σχήμα 1.2: Ζυγές διαστάσεις εικόνας

2 Από DNG σε RGB

Η όλη διαδικασία της μετατροπής χωρίζεται σε 4 κομμάτια:

1. Δειγματοληψία και αλλαγή μεγέθους
2. Ρύθμιση του *white balance*
3. Demosaicing
4. Color space conversions

Το πρώτο και το τέταρτο είναι κοινό για όλα τα *Bayer patterns*, το δεύτερο ξεχωριστό για το καθένα ενώ στο *demoslicing* υπάρχουν ομοιότητες που εκμεταλλευόμαστε. Υπάρχει ο κατάλληλος έλεγχος στην αρχή του κώδικα και συναρτήσεις για κάθε περίπτωση, αλλά ως επί των πλείστων θα παρουσιάζεται ο κώδικας για το *RGGB pattern* μιας και είναι το πλέον δημοφιλέστερο.

2.1 Δειγματοληψία

Η δειγματοληψία συμβαίνει ανεξάρτητα του *Bayer pattern*, αλλά για κάθε είδος αισθητήρα ξεχωριστά. Άσχετα με το πιο χρώμα αντιπροσωπεύουν σε κάθε μοτίβο, γνωρίζουμε πως οι αισθητήρες σε μονές σειρές και στήλες είναι ίδιοι, παρόμοια οι αισθητήρες σε ζυγές σειρές και στήλες κτλ. Έτσι, μπορούμε με κατάλληλα βήματα και επιλογές σειρών και στηλών να δειγματοληπτήσουμε την `rawim` ανεξάρτητα του *Bayer pattern*. Ο μόνος περιορισμός που υπάρχει είναι καινούριες οι διαστάσεις που χρησιμοποιούνται να είναι ζυγός αριθμός, αλλά δεν χρειάζεται να τηρείται το *aspect ratio* ή να είναι ακέραια (υπο)πολλαπλάσια των αρχικών. Χρησιμοποιώντας βήμα $\frac{M_0}{M} * 2$ (και αντίστοιχα για τις στήλες) δειγματοληπτούμε σε συγκεκριμένα σημεία, διατηρώντας το ίδιο μοτίβο. Το $*2$ χρησιμοποιείται λόγω της διάταξης του *Bayer pattern*, αφού σε κάθε σειρά (και στήλη) υπάρχουν δύο χρώματα και το καθένα με βήμα 2. Στο Σχήμα 2.1 φαίνεται η απλή περίπτωση της διαδικασίας αυτής, όπου το $\frac{M_0}{M}$ είναι ακέραιος. Σε άλλη περίπτωση, χρείαζεται να γίνουν οι αντίστοιχες στρογγυλοποιήσεις και έλεγχοι για *edge cases* οπότε δεν φαίνεται εδώ για συντομία. Οι διασχίσεις του πίνακα είναι ανεστραμμένες σε ανάλογα σημεία ώστε οι γωνιακές τιμές της αρχικής εικόνας να ισοδυναμούν με αυτές της δειγματοληπτημένης.

Οποιαδήποτε επεξεργασία μετά συμβαίνει στο νέο πλέγμα MxN .

2.2 White balance

Παρόμοια με την δειγματοληψία, το μοτίβο κάθε χρώματος χρησιμοποιείται για την εξαγωγή της κάθε χρωματικής συνιστώσας και την δημιουργία μιας μάσκας για την ρύθμιση του *white balance*. Στο Σχήμα 2.2 φαίνονται τα βήματα που χρησιμοποιούνται για την αρχικοποίηση του πίνακα με στόχο τον πολλαπλασιασμό κάθε χρώματος με τον αντίστοιχο συντελεστή. Το πράσινο χρώμα μένει αναλλοίωτο.

```

% step of sampling rate in each dimension
stepM = M0 / M * 2;
stepN = N0 / N * 2;

sampled = zeros(M, N);

% % sample each color
% two ways because when M0 / M and N0 / N is integer it is faster
if mod(M0, M) == 0 && mod(N0, N) == 0
    % reverse order of sampling to make sure that
    % the edges of the grid is literally the same value
    sampled(1:2:end, 1:2:end) = rawim(1:stepM:end, 1:stepN:end);
    sampled(1:2:end, end:-2:2) = rawim(1:stepM:end, end:-stepN:2);
    sampled(end:-2:2, 1:2:end) = rawim(end:-stepM:2, 1:stepN:end);
    sampled(2:2:end, end:-2:2) = rawim(2:stepM:end, end:-stepN:2);
else
    % second way is the classic with 2 fors
    % it essentially does the same thing but with an decimal step
    % which is rounded to the nearest censor of the same color

```

Σχήμα 2.1: Δειγματοληψία χρωμάτων

```

% % white balance
% mask with the size of image
% leave green as it is
wbmask = zeros(M, N);
wbmask(1:2:end, 1:2:end) = wbcoeffs(1);
wbmask(2:2:end, 2:2:end) = wbcoeffs(3);
red = red .* wbmask;
blue = blue .* wbmask;

wbmask = zeros(M, N);
wbmask(2:2:end, 2:2:end) = wbcoeffs(1);
wbmask(1:2:end, 1:2:end) = wbcoeffs(3);
red = red .* wbmask;
blue = blue .* wbmask;

wbmask = zeros(M, N);
wbmask(2:2:end, 1:2:end) = wbcoeffs(1);
wbmask(1:2:end, 2:2:end) = wbcoeffs(3);
red = red .* wbmask;
blue = blue .* wbmask;

wbmask = zeros(M, N);
wbmask(1:2:end, 2:2:end) = wbcoeffs(1);
wbmask(2:2:end, 1:2:end) = wbcoeffs(3);
red = red .* wbmask;
blue = blue .* wbmask;

```

Σχήμα 2.2: White balance correction σε RGGB, BGGR, GBRG και GRBG αντίστοιχα

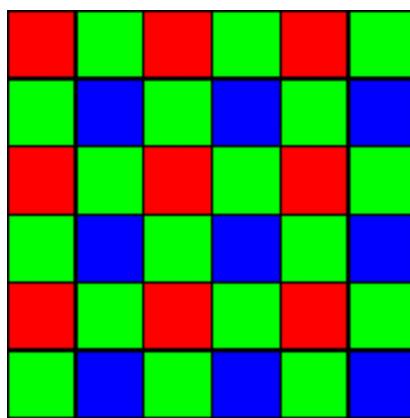
2.3 Demosaicing

Οι τρεις, δειγματοληπτημένοι και διορθωμένοι πίνακες που λαμβάνουμε έχουν τιμή μόνο στα σημεία που υπάρχουν στην πραγματικότητα αισθητήρες για το αντίστοιχο χρώμα. Η δημιούργια των υπόλοιπων τιμών γίνεται με βάση αυτές, όπως περιγράφεται παρακάτω.

2.3.1 Nearest neighbor

Για την συμπλήρωση των κενών με την μέθοδο του κοντινότερου γείτονα ορίζουμε αυθαίρετα για κάθε είδος κενού έναν αισθητήρα του χρώματος που θέλουμε να συμπληρώσουμε ως κοντινότερο. Αυτό γίνεται με βάση το αντίστοιχο *Bayer pattern*. Στο Σχήμα 2.3 βλέπουμε σαν παράδειγμα το *RGGB*. Όπως έχουμε ήδη αναφέρει κατά την ανάγνωση της εικόνας, είναι σημαντικό οι διαστάσεις της να είναι ζυγές ώστε στα άκρα να διατηρείται αναλλοίωτο το μοτίβο των χρωμάτων. Δηλαδή, στην περίπτωση του *RGGB*, θεωρούμε ότι είναι αδύνατο μία μονή σειρά να τελειώνει σε κόκκινο αισθητήρα ή μια ζυγή σε πράσινο. Με αυτή την υπόθεση- η οποία όπως είπαμε ισχύει σχεδόν πάντα ή μπορεί να επιτευχθεί με ευκολία χωρίς ιδιαίτερο κόστος πληροφορίας- υπάρχουν πάντα και παντού τέλειες τετράδες *RGGB* ώστε να μην χρείαζεται να αλλάζουμε τον αισθητήρα που θεωρούμε κοντινότερο. Συγκεκριμένα, κοιτάζοντας το πάνω αριστερά, 2x2 τετράγωνο, θεωρούμε:

- Για τον κόκκινο αισθητήρα, ως κοντινότερο πράσινο αισθητήρα τον *East (E)* και ως κοντινότερο μπλε τον *SE*.
- Για τον πράσινο της πάνω σειράς, ως κοντινότερο κόκκινο τον *W* και ως κοντινότερο μπλε τον *S*.
- Για τον πράσινο της κάτω σειράς, ως κοντινότερο κόκκινο τον *N* και ως κοντινότερο μπλε τον *E*.
- Για τον μπλε, ως κοντινότερο πράσινο τον *W* και ως κοντινότερο κόκκινο τον *NW*.



Σχήμα 2.3: RGGB Bayer pattern

Είναι προφανές πως όταν δεν είμαστε στα όρια της φωτογραφίας έχουμε διάφορες επιλογές τις οποίες δεν χρησιμοποιούμε για να μην αλλάζουμε μοτίβο επιλογής. Στην συνέχεια, στο στρώμα κάθε χρώματος (π.χ. *R*) συμπληρώνουμε τις θέσεις που είχε αισθητήρες των άλλων δύο (*G* και *B*) με βάση τον κοντινότερο γείτονα που ορίσαμε. Για παράδειγμα, στον πίνακα *red* στις θέσεις που αναλογούν σε έναν μπλε αισθητήρα (ζυγές γραμμές και στήλες) αντιγράφουμε την *NW* τιμή (μονές γραμμές και στήλες). Αυτό μεταφράζεται σε κώδικα σε συγκεκριμένα βήματα διάσχισης του πίνακα, παρόμοια με την δειγματοληψία και

το *white balance*. Στο Σχήμα 2.4 φαίνεται ένα παράδειγμα για την μεταφορά των τιμών των κόκκινων αισθητήρων σε θέσεις πράσινων και μπλε. Παρόμοια γίνεται και στο *GBRG* αλλάζοντας μόνο τα βήματα μεταξύ τους, ενώ στα *BGGR* και *GRBG* δεν χρειάζεται να υλοποιηθεί νέα συνάρτηση αφού έχουν την ίδια σχετική θέση μεταξύ τους οι αισθητηρές, με μόνη διαφορά πως οι κόκκινοι και οι μπλε είναι ανάποδα. Αυτό μπορεί να διορθωθεί με μια απλή "ανταλλαγή" των τιμών *R* και *B* μετά τον υπολογισμό όλων των σημείων (Σχήμα 2.5). Η διαδικασία αυτή ολοκληρώνει την μετατροπή σε *RGB* με την μέθοδο του κοντινότερου γείτονα.

```
% % fill red on blue censors spots
% nearest neighbor is considered NW value
red(2:2:end, 2:2:end) = red(1:2:end, 1:2:end);

% % fill red on green censors spots
% nearest neighbor is considered W and N value
red(1:2:end, 2:2:end) = red(1:2:end, 1:2:end);
red(2:2:end, 1:2:end) = red(1:2:end, 1:2:end);
```

Σχήμα 2.4: Συμπλήρωση των τιμών του κόκκινου χρώματος σε RGGB

```
% bggr is essentially the same as rggb with r and b flipped
Ccam = nearestRGGB(sblue, sgreen, sred);

sred = Ccam(:, :, 3);
Ccam(:, :, 3) = Ccam(:, :, 1);
Ccam(:, :, 1) = sred;
```

Σχήμα 2.5: Διόρθωση από RGGB σε BGGR Bayer pattern

2.3.2 Bilinear interpolation

Στην μέθοδο της διγραμμικής παρεμβολής λαμβάνουμε υπόψιν τις τιμές των 8-γειτόνων για την παρεμβολή του χρώματος στον αισθητήρα που ψάχνουμε. Εφόσον οι σχετικές θέσεις είναι συγκεκριμένες και όλοι οι συμμετέχοντες αισθητήρες έχουμε απόσταση δύο μεταξύ τους, η διγραμμική παρεμβολή εκφυλίζεται απλώς στον μέσο όρο των αισθητήρων του χρώματος που θέλουμε να συμπληρώσουμε. Αυτό μπορεί να υλοποιηθεί χρησιμοποιώντας την συνάρτηση *imfilter*, ψεωρώντας *zero-padding* και πολύ απλές μάσκες για κάθε χρώμα. Να σημειωθεί σε αυτό το σημείο πως οι μάσκες αυτές καταλήγουν ίδιες και για τα τέσσερα *Bayer patterns* οπότε και δεν υπάρχει λόγος υλοποίησης διαφορετικών συναρτήσεων για κάθε περίπτωση. Συγχερικούμενα:

- Για θέση κόκκινου αισθητήρα, οι μπλε που θα χρησιμοποιηθούν για την διγραμμική παρεμβολή στην ίδια θέση στο στρώμα *B* θα είναι οι *NW*, *NE*, *SE* και *SW*. Αντίστοιχα, για το πράσινο θα είναι οι *N*, *E*, *S* και *W*.
- Για πράσινο αισθητήρα, και οι μπλε και οι κόκκινα είναι είτε *W*, *E* είτε *N*, *S*.
- Για μπλε αισθητήρα, οι κόκκινοι είναι *NW*, *NE*, *SE* και *SW*, ενώ οι πράσινοι *N*, *E*, *S* και *W*.

Με βάση τις παραπάνω παρατηρήσεις, παρόλο που τα χρώματα αλλάζουν θέσεις από *pattern* σε *pattern* οι μάσκες που χρησιμοποιούνται παραμένουν ίδιες. Στο Σχήμα 2.6 φαίνεται η υλοποίηση των παραπάνω.

```

% fill red on blue spots
redBlue = imfilter(red, [1 0 1; 0 0 0; 1 0 1] / 4);

% fill red on green spots
redGreen1 = imfilter(red, [1 0 1;] / 2);
redGreen2 = imfilter(red, [1; 0; 1] / 2);
redGreen = redGreen1 + redGreen2;

red = red + redBlue + redGreen;

% green filling is the same for both red and blue
greenRedBlue = imfilter(green, [0 1 0; 1 0 1; 0 1 0] / 4);

green = green + greenRedBlue;

% fill blue on red spots
blueRed = imfilter(blue, [1 0 1; 0 0 0; 1 0 1] / 4);

% fill blue on green spots
blueGreen1 = imfilter(blue, [1 0 1;] / 2);
blueGreen2 = imfilter(blue, [1; 0; 1] / 2);
blueGreen = blueGreen1 + blueGreen2;

blue = blue + blueRed + blueGreen;

Ccam = zeros(M, N, 3);
Ccam(:, :, 1) = red;
Ccam(:, :, 2) = green;
Ccam(:, :, 3) = blue;

```

Σχήμα 2.6: Διγραμμική παρεμβολή

2.4 Color space conversions

Με την ολοκλήρωση της παρεμβολής για την εύρεση όλων των χρωμάτων παίρνουμε την εικόνα C_{cam} . Για την μετατροπή στην C_{XYZ} παίρνουμε τον ανάστροφο του πίνακα $T_{XYZ \rightarrow Cam}$, τον οποίο είχαμε αποθηκέψει από τα `meta_info`. Αντίστοιχα, για την μετατροπή στον C_{linear} χρησιμοποιείται ο $T_{XYZ \rightarrow RGB}$. Ακολουθεί, μια τυπική διόρθωση της φωτεινότητας και χρησιμοποιείται η προσσέγιση για την παραγωγή της τελικής εικόνας C_{sRGB} . Πριν τους μετασχηματισμούς κάθισε πίνακας κανονικοποιείται ώστε οι γραμμές του να ανθροίζουν στην μονάδα για να υπάρχει διατήρηση του άσπρου χρώματος από το ένα *colorspace* στο άλλο. Μετά από κάθισε μετασχηματισμό η εικόνα ελέγχεται και διορθώνεται για τυχόν τιμές εκτός του $[0, 1]$.

3 Αποτελέσματα

Αρχικά, παρατίθονται οι εικόνες C_{cam} , C_{XYZ} , C_{linear} και C_{sRGB} για το .dng αρχείο που δίνεται (Σχήματα 3.1-3.4). Οι εικόνες αυτές είναι με τις αρχικές διαστάσεις $4000x6000$ και με την μέθοδο του κοντινότερου γείτονα. Βλέπουμε πως αρχικά η εικόνα είναι πιο σκοτεινή και τα χρώματα λίγο διαφορετικά αλλά παιρνώντας σιγά σιγά στο σωστό *color space* που είναι κατάλληλο για την προβολή της εικόνας αυτά διορθώνονται. Η τελική εικόνα είναι στο Σχήμα 3.4. Η εικόνα που θα λάβουμε σε αυτό το σημείο εξαρτάται και σε μεγάλο βαθμό από τις διορθώσεις που θα γίνουν όταν μεταβαίνουμε από την C_{linear} στην τελική.

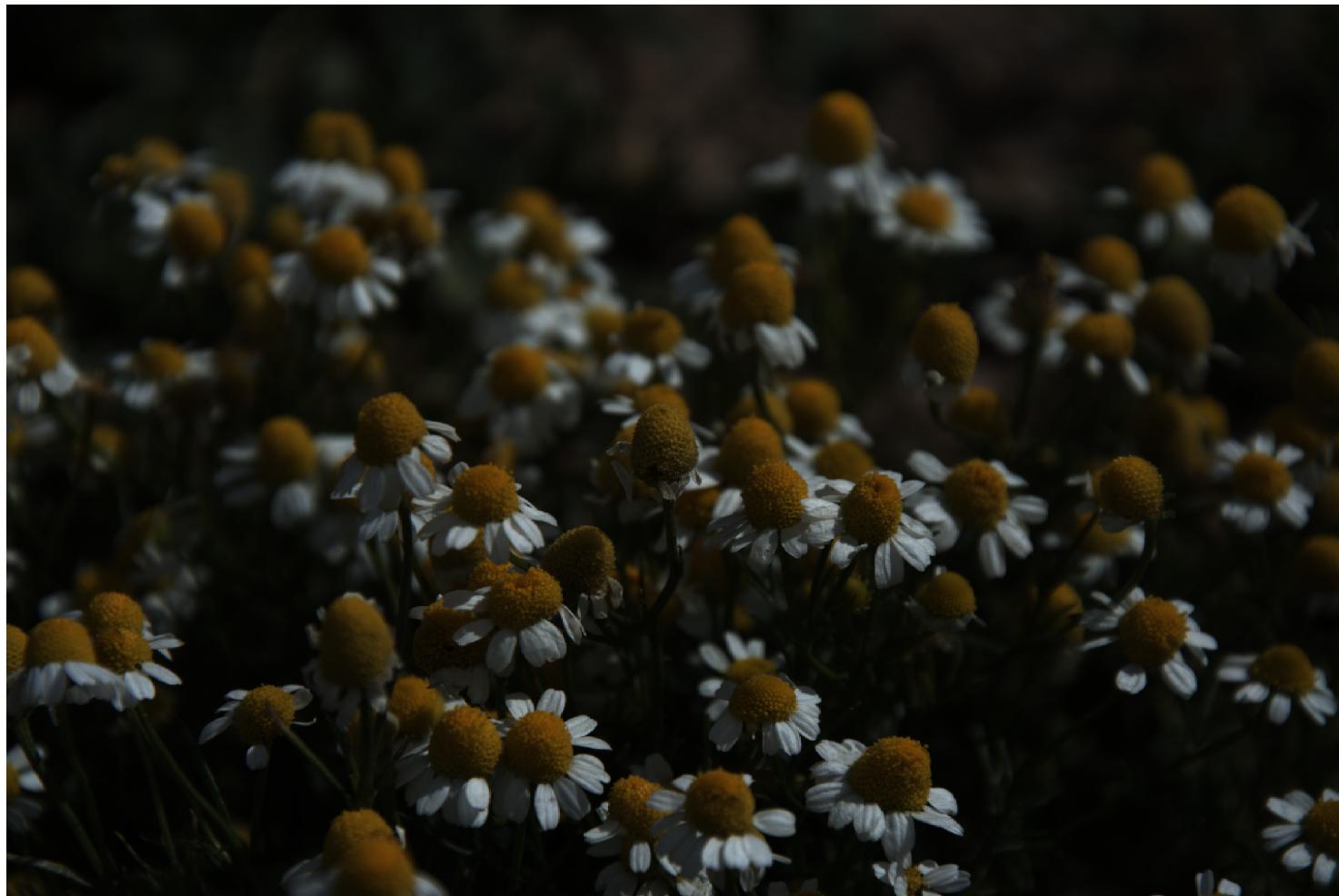
Για καλύτερη σύγκριση, στα Σχήματα 3.5-3.8 φαίνονται τα ιστογράμματα κάθε εικόνας για κάθε κανάλι ξεχωριστά. Στην C_{cam} και στην C_{XYZ} (Σχήματα 3.5-3.6) όλα τα κανάλια είναι συγκεντρωμένα στο κάτω άκρο, όπως περιμέναμε κι όλας λόγω της σκοτεινότητας των εικόνων. Περνώντας στην C_{linear} (Σχήμα 3.7) υπάρχει μια γενική αύξηση αλλά πολύ μικρή. Στην αντίστοιχη εικόνα, ενώ έχει αυξηθεί λίγο το *contrast* δεν υπάρχει ιδιαίτερη βελτίωση στην φωτεινότητα ούτε στην ένταση των χρωμάτων. Τέλος, στο ιστόγραμμα της C_{sRGB} (Σχήμα 3.8) οι τιμές όλων των χρωμάτων έχουν αυξηθεί (διόρθωση της φωτεινότητας), ενώ τα ύψη που υπήρχαν συγκεντρωμένα γύρω από λίγες τιμές έχουν κατανευμηθεί σε μεγαλύτερο διάστημα (*gamma correction*) έχοντας σαν αποτέλεσμα την αύξηση του *contrast*.

Στο Σχήμα 3.9 βλέπουμε την τελική εικόνα αλλά με διγραμμική παρεμβολή. Λόγω της πυκνής πληροφορίας που έχουμε αφού η εικόνα μας είναι $4000x6000$ δεν υπάρχει διαφορά μεταξύ των δύο μεθόδων και μετατρέπουν την είκονα σε *RGB* επιτυχώς. Στο Σχήμα 3.10 είναι η ίδια αλλά με διαστάσεις $7000x9000$.

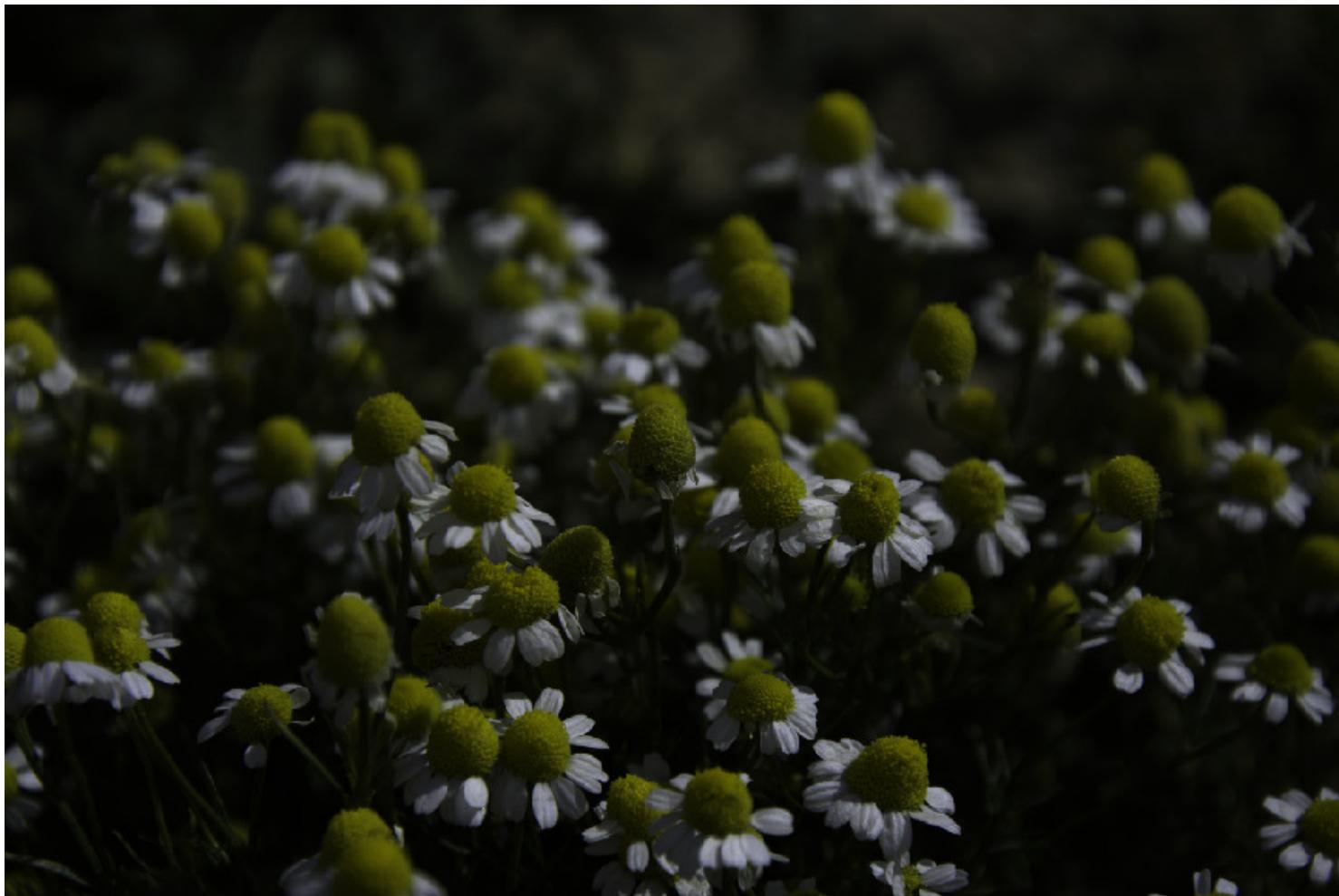
Στο Σχήμα 3.11 χρησιμοποιούμε για εισόδους διαστάσεις $2500x3750$ και διγραμμική παρεμβολή. Υπάρχει μια μικρή διαφορά από την μεγαλύτερη εικόνα αλλά δεν είναι ιδιαίτερα εμφανής ακόμη. Άμα συνεχίζουμε την μείωση των διαστάσεων η διαφορά γίνεται πλεόν φανερή (Σχήμα 3.12), ενώ αμάχρηστη συνέχιση της διαστάσεων $1238x1238$ με την μέθοδο του κοντινότερου γείτονα αρχίζουμε να εντοπίζουμε και λάθη. Για παράδειγμα, κάποια *pixel* που είναι στην άκρη των λουλουδιών δεν είναι τόσο κίτρινα αλλά πράσινα ή κάποιο άλλο χρώμα γύρω από το λουλούδι (λευκό).

Τέλος, για πληρότητα, παρατίθενται στα Σχήματα 3.14-3.16 οι εικόνες που παίρνουμε έαν χρησιμοποιήσουμε την *dng2rgb* με λάθος *Bayer pattern* (άλλο από αυτό της κάμερας με την οποία τραβήχτηκε η φωτογραφία). Στην περίπτωση του *BGGR* παίρνουμε διαφορετικό χρώμα στα λουλούδια ενώ όλα τα χρώματα είναι πιο "χρύα". Αυτό συμβαίνει λόγω της ανάποδης τοποθέτησης που έχουν οι κόκκινοι και οι μπλε αισθητήρες σε σχέση με το *RGGB*. Στις άλλες εικόνες δεν υπάρχει τόσο απλή εναλλαγή χρωμάτων αφού όλα τα χρώματα έχουν αλλάξει θέσεις και δεν έχουμε ομαλή μετάβαση.

∞



$\Sigma\chi\nu\alpha$ 3.1: C_{cam} , 4000x6000



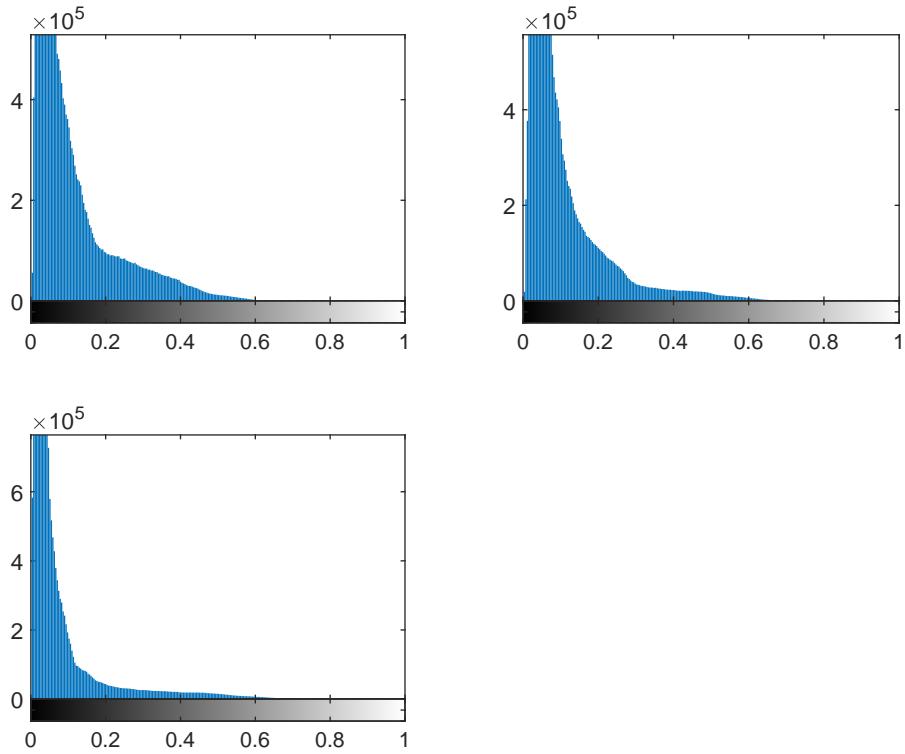
$\Sigma\chi\nu\mu\alpha$ 3.2: C_{XYZ} , 4000x6000



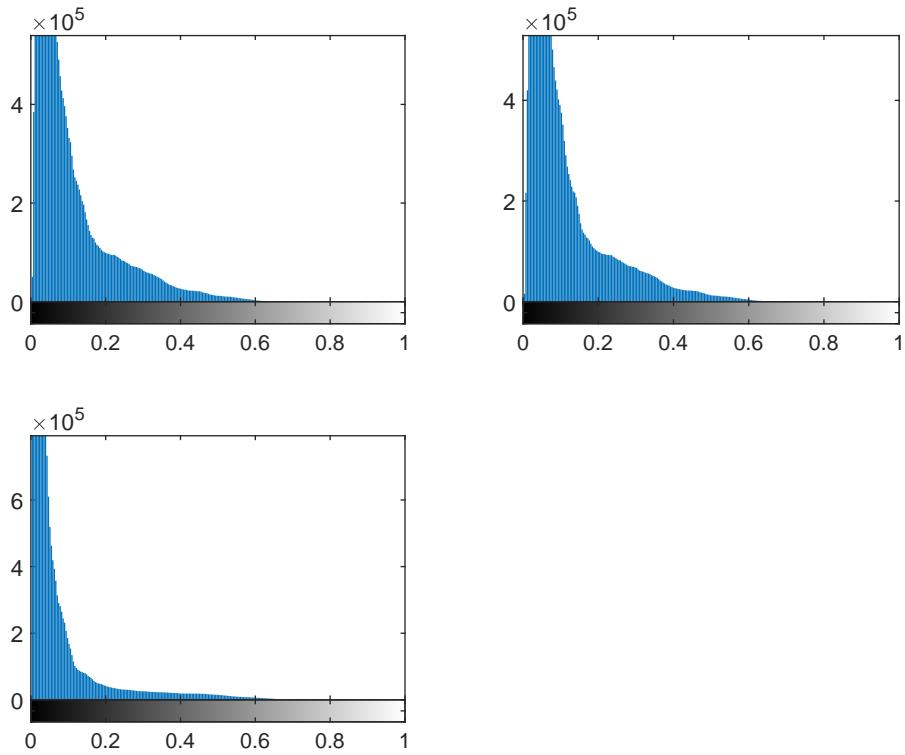
$\Sigma\chi\nu\alpha$ 3.3: C_{linear} , 4000x6000



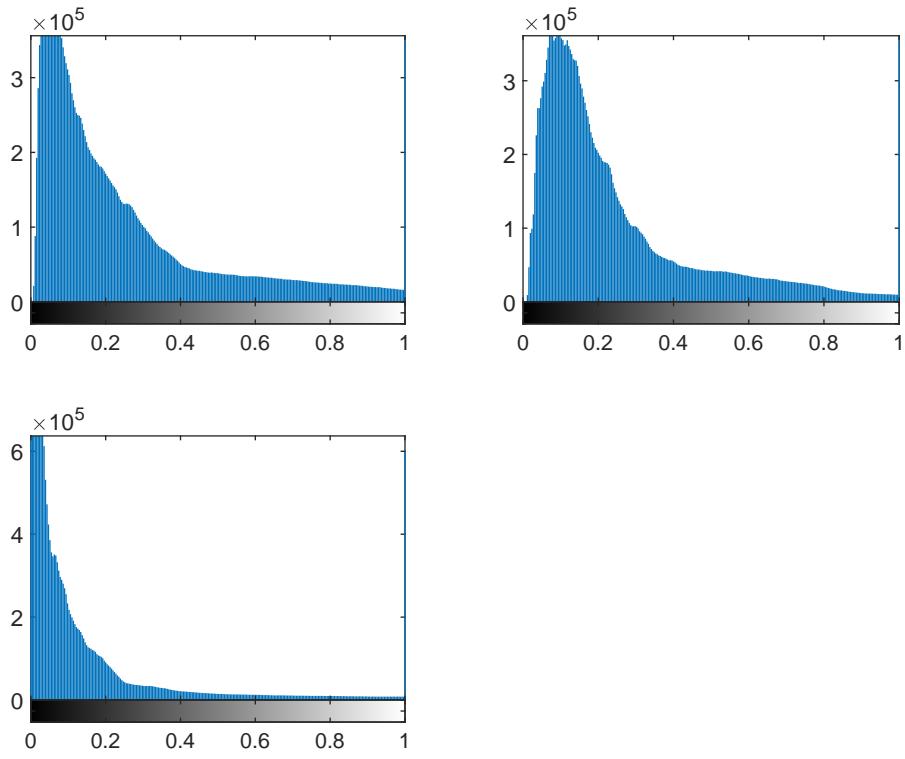
Σχήμα 3.4: C_sRGB 4000x6000



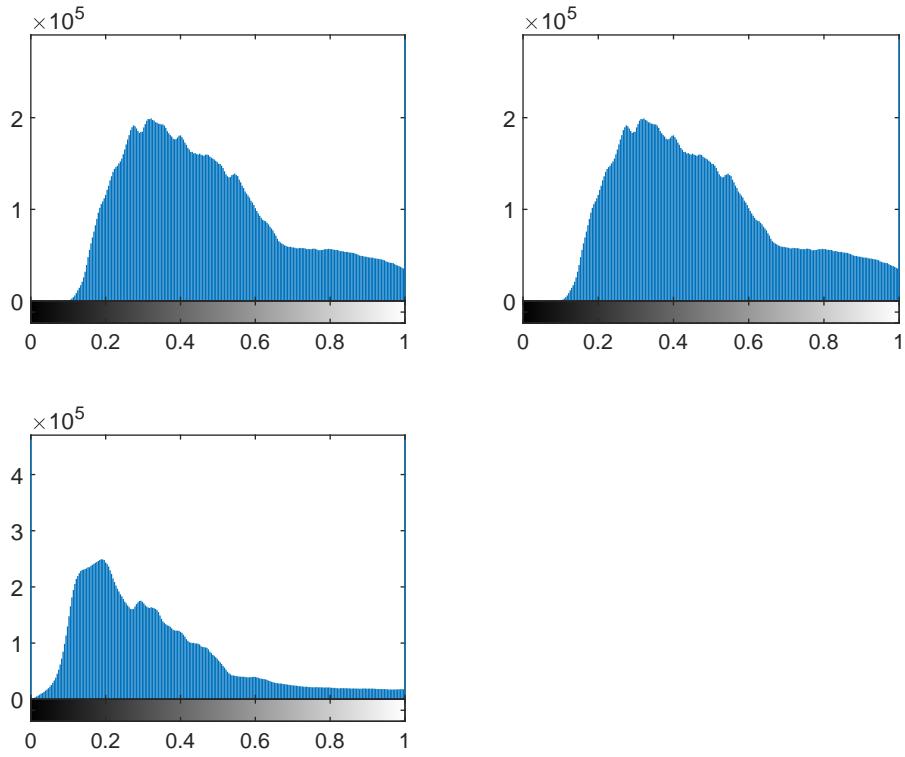
Σχήμα 3.5: Ιστόγραμμα της C_{cam} για τα κανάλια R, G, B αντίστοιχα



Σχήμα 3.6: Ιστόγραμμα της C_{XYZ} για τα κανάλια R, G, B αντίστοιχα



$\Sigma\chi\rho\mu\alpha$ 3.7: Ιστόγραμμα της C_{linear} για τα κανάλια R, G, B αντίστοιχα



$\Sigma\chi\rho\mu\alpha$ 3.8: Ιστόγραμμα της C_{sRGB} για τα κανάλια R, G, B αντίστοιχα



$\Sigma\chi\mu\alpha$ 3.9: C_{sRGB} , bilinear, 4000x6000



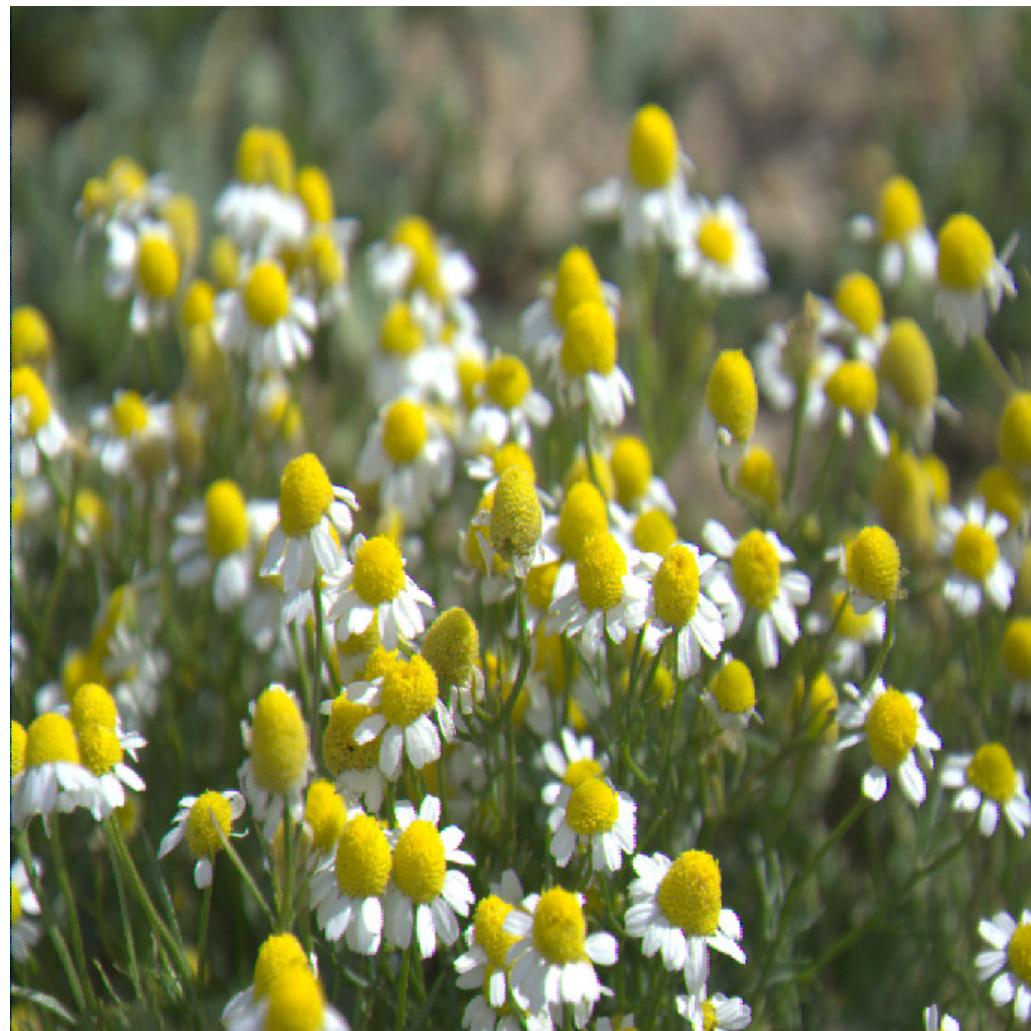
$\Sigma\chi\nu\alpha$ 3.10: C_{sRGB} , bilinear, 7000x9000



$\Sigma\chi\nu\alpha$ 3.11: C_{sRGB} , bilinear, 2500x3750



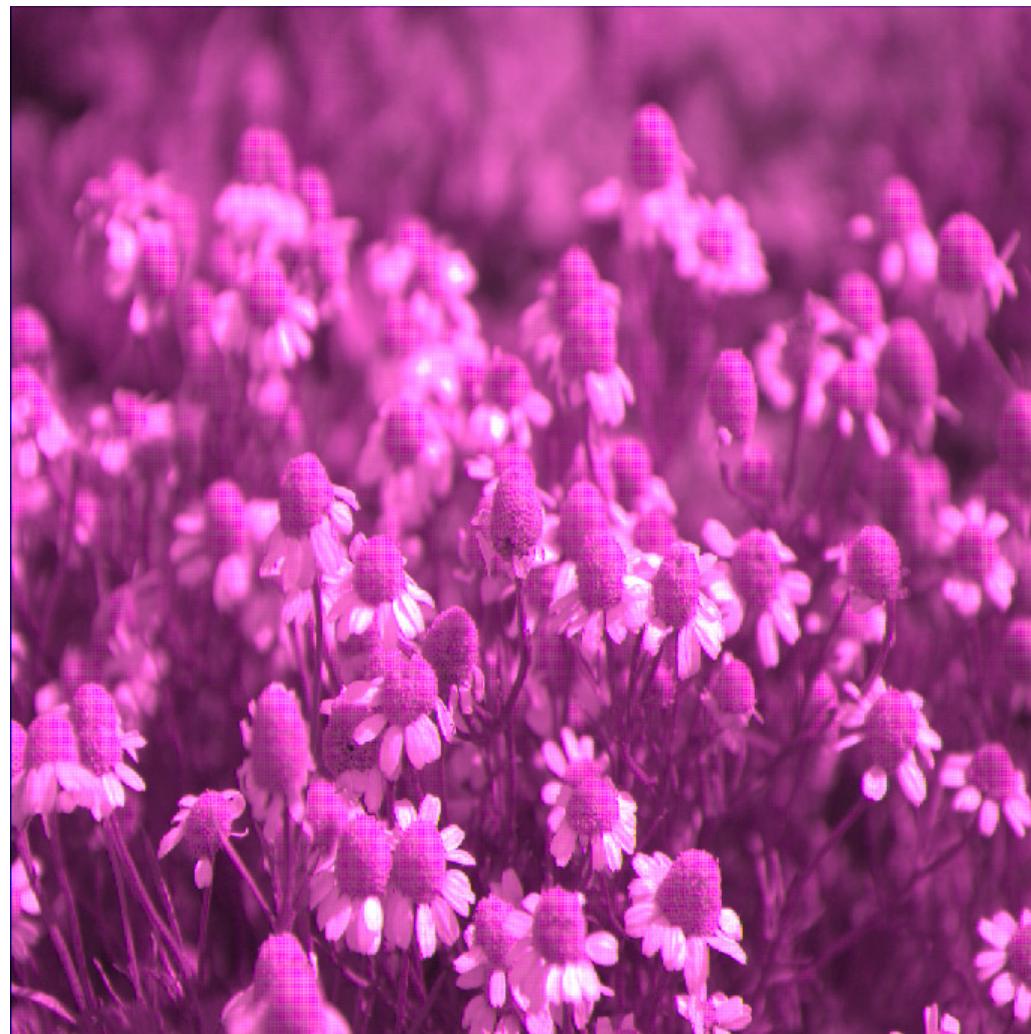
$\Sigma\chi\nu\alpha$ 3.12: C_sRGB , bilinear, 1238x1238



$\Sigma\chi\nu\alpha$ 3.13: C_sRGB , nearest, 1238x1238



$\Sigma\chi\acute{\eta}\mu\alpha$ 3.14: C_{sRGB} , $BGGR$, 1238×1238



$\Sigma\chi\gamma\mu\alpha$ 3.15: C_{sRGB} , $GBRG$, 1238×1238



$\Sigma\chi\gamma\mu\alpha$ 3.16: C_{sRGB} , GRBG, 1238x1238