Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Ψηφιακή Επεξεργασία Εικόνας

Άνοιξη 2021 - 8ο Εξάμηνο

-Εργασία 1-

Bayer, downsampling, quantization & PPM representation

Ονοματεπώνυμο: Τσούσης Παναγιώτης

ΑΕΜ: 9590

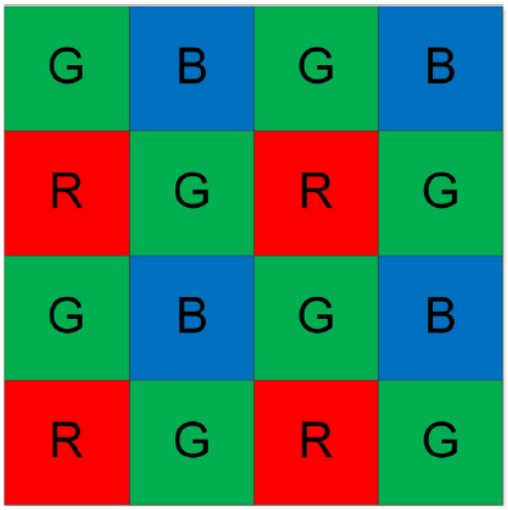
Email: pitsousis@ece.auth.gr

Πίνακας περιεχομένων

1 - Φίλτρο Bayer

1.1 - Στόχος της άσκησης

Ο στόχος της άσκησης είναι να ανακατασκευαστεί μια εικόνα που έχει ληφθεί από μια φωτογραφική μηχανή που κάνει χρήση του φίλτρου Bayer. Συγκεκριμένα, μετά από πειραματισμό, διαπιστώθηκε πως στην εργασία αυτή το φίλτρο έχει μια διάταξη στους αισθητήρες χρώματός της γνωστή ως 'διάταξη gbrg' και είναι αυτή που φαίνεται στην εικόνα 1.



Εικόνα 1: Bayer filter - gbrg pattern

Με δεδομένο αυτό και τα στοιχεία φωτεινότητας που δόθηκαν στον πίνακα march.mat, έχουμε τη δυνατότητα να ανακατασκευάσουμε την εικόνα από τα δείγματα της κάμερας. Θα χρησιμοποιήσουμε μια μέθοδο averaging , δηλαδή το συγκεκριμένο χρώμα που θα πρέπει να έχει το pixel (red, green, blue) να είναι ο μέσος όρος των γειτονικών του με το ίδιο χρώμα.

1.2 - Kernels, συνέλιξη και τελική εικόνα

Σε μια εικόνα που έχει ληφθεί με τον τρόπο που περιεγράφηκε παραπάνω, είναι εμφανές ότι ενώ θα έπρεπε να έχουμε 3 τιμές για κάθε pixel (1 για κάθε χρώμα), έχουμε μόνο μία. Για το λόγο αυτό πρέπει να κάνουμε κάποια υπόθεση για να συμπληρώσουμε τις άλλες 2 τιμές που λείπουν.

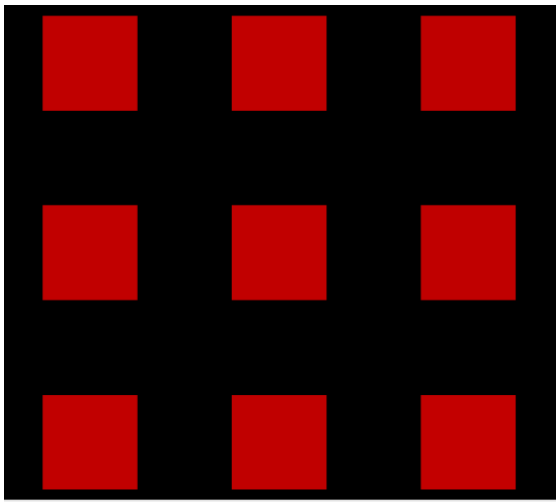
Για το λόγο αυτό, δημιουργούμε μια συνάρτηση στο Matlab, την `function xrgb = bayer2rgb(xb).` Η συνάρτηση αυτή παίρνει ως είσοδο τον αρχικό πίνακα `xb` που είναι τα δείγματα από τη φωτογραφική μηχανή και επιστρέφει έναν τρισδιάστατο πίνακα `xrgb` που είναι η αναπαράσταση της εικόνας ως truecolor image.

Σε πρώτο στάδιο χωρίζουμε τον πίνακα `xb` σε 3 δισδιάστατους πίνακες, που ονομάζουμε `R`, `G` και `B`. Ο κάθε πίνακας περιέχει τις τιμές του αντίστοιχου χρώματος στις θέσεις που βρίσκονταν στον αρχικό πίνακα, ενώ στις υπόλοιπες θέσεις είναι 0. Αυτό είναι δυνατό, διότι το φίλτρο Bayer έχει πολύ συγκεκριμένη δομή.

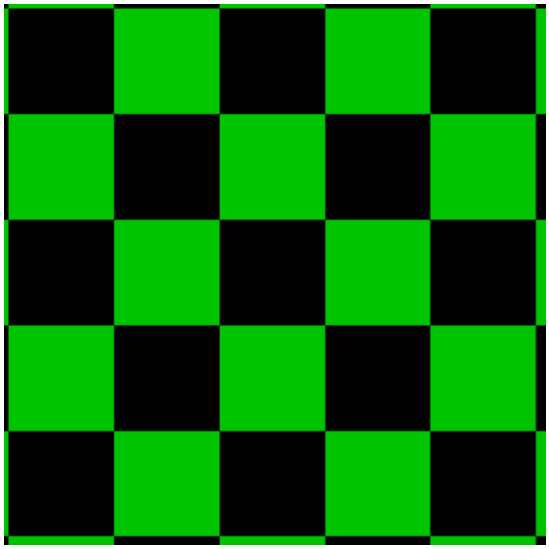
Στη συνέχεια, θέλουμε να εφαρμόσουμε μία συνέλιξη του κάθε πίνακα που δημιουργήσαμε με μια μάσκα (kernel), ώστε να βρούμε την τιμή των pixels με τιμή 0 για το συγκεκριμένο χρώμα. Το ερώτημα για την ακριβή δομή της μάσκας λύνεται αν κάνουμε 2 βασικές υποθέσεις:

- το χρώμα ενός pixel εξαρτάται από την άμεση γειτονία του, δηλαδή από τα pixels 1 θέση πάνω, κάτω, αριστερά, δεξιά και διαγώνια του pixel που μας ενδιαφέρει
- το χρώμα του παραπάνω pixel είναι *το άθροισμα των τιμών των pixel της γειτονιάς του, διαιρούμενο κατά το πλήθος των pixel της γειτονιάς του που έχουν* ***μη μηδενική τιμή.***

Με την παραπάνω λογική, δημιουργούνται 2 kernels μεγέθους 3x3, ένα για τους πίνακες `R` και `B` (`kernel_rb`) και ένα για τον πίνακα `G` (`kernel_g`). Η διαφορά προκύπτει από το γεγονός ότι οι πρώτοι 2 πίνακες έχουν το ίδιο μοτίβο στις θέσεις των ήδη χρωματισμένων pixels ενώ ο τρίτος διαφέρει. Αυτό φαίνεται και στις εικόνες 2 και 3.



Εικόνα 2: Διάταξη red/blue σε 5x5



Εικόνα 3: Διάταξη green σε 5x5

Με βάση τις παραπάνω εικόνες μπορούμε να διερευνήσουμε τη μορφή των kernels.

1.2.1 - Kernel_rb

Έχουμε 2 περιπτώσεις:

- Το pixel που μας ενδιαφέρει (βρίσκεται στο στοιχείο (2,2) του kernel) έχει ήδη μη μηδενική τιμή.
- Το pixel που μας ενδιαφέρει έχει μηδενική τιμή.

Στην πρώτη περίπτωση, πρέπει μετά τη συνέλιξη να μην έχει αλλάξει το χρώμα του. Με δεδομένο ότι τα γειτονικά pixels έχουν μηδενική τιμή, σε αυτήν την περίπτωση δε μας ενδιαφέρει τι τιμές θα μπουν στα γειτονικά κελιά του (2,2) στο kernel_rb. Άρα, αναγκαστικά, τελικά, στο κελί (2,2) του kernel_rb θα μπει η τιμή 1, ώστε η έξοδος να είναι η ίδια τιμή.

Στη δεύτερη περίπτωση το pixel που μας ενδιαφέρει έχει μηδενική τιμή. Με βάση τη γεωμετρία που φαίνεται στην εικόνα 2, το pixel μπορεί να είναι στο κέντρο ενός μαύρου σταυρού. Έτσι, έχουμε 4 γειτονικά με χρώμα, άρα στις «γωνίες» του kernel_rb πρέπει να μπει ο όρος ¼. Το κεντρικό 1 που βάλαμε πριν δε μας επηρεάζει, καθώς η τιμή του ίδιου του pixel είναι ήδη 0. Τέλος, το pixel μπορεί να είναι σε κάποια από τις κορυφές του μαύρου σταυρού. Στην περίπτωση αυτή έχει 2 γειτονικά (πάνω/κάτω ή αριστερά/δεξιά) pixels, οπότε στις 4 πλευρές του kernel_rb πρέπει να μπει ο όρος ½. Τα υπόλοιπα κελιά που βάλαμε πριν πάλι δε μας επηρεάζουν, καθώς οι τιμές τους σε αυτήν την περίπτωση είναι πάλι 0.

Με την παραπάνω ανάλυση, προκύπτει το φίλτρο kernel_rb:

$$kernel\_rb = \frac{1}{4}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

1.2.2 - Kernel_g

Με παρόμοια μέθοδο με αυτή που ακολουθήσαμε παραπάνω, προκύπτει πως το φίλτρο kernel_g είναι:

$$kernel\_g = \frac{1}{4}\begin{bmatrix}0 & 1 & 0\\ 1 & 4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

1.2.3 - Συνέλιξη

Αφού έχουμε βρει τα kernels, μπορούμε πλέον να εφαρμόσουμε τη συνέλιξη `conv2` του Matlab των πινάκων `R, G και B` με τα αντίστοιχα φίλτρα τους και να δώσουμε σε κάθε κελί των πινάκων τους μια τιμή. Μετά την εφαρμογή αυτή, προκύπτουν 3 νέοι πίνακες, οι `R_conv, B_conv και G_conv` .

1.2.4 - Σύνθεση τελικής εικόνας

Πλέον, το μόνο που μένει να κάνουμε, είναι να συνθέσουμε τον πίνακα `xrgb` που είναι απλά μια πρόσθεση των δισδιάστατων πινάκων `R_conv, B_conv και G_conv` σε ένα τρισδιάστατο πίνακα.

Τέλος, η συνάρτηση `function xrgb = bayer2rgb(xb)` επιστρέφει τον πίνακα αυτό.

1.3 - Αποτελέσματα και συμπεράσματα

Με την εφαρμογή των παραπάνω σε κώδικα Matlab και δίνοντας ως είσοδο τα δεδομένα `march.mat`, μπορούμε πλέον να βγάλουμε αποτελέσματα και συμπεράσματα.

1.3.1 - Αποτελέσματα

Χωρίς επεξεργασία της εικόνας (αποτέλεσμα από test που έγιναν) η raw εικόνα που τραβάει η κάμερα φαίνεται στην εικόνα 4.

Εικόνα 4: Προβολή raw εικόνας από την κάμερα

Μετά την εφαρμογή του αλγορίθμου που κατασκευάστηκε, με είσοδο την παραπάνω εικόνα, η εικόνα που παράγεται φαίνεται στην εικόνα 5.

Εικόνα 5: Εικόνα μετά την εφαρμογή του αλγορίθμου bayer2rgb

Σημειώνεται ότι κατά την προβολή στο Matlab, παρουσιάστηκε η παρακάτω ειδοποίηση:

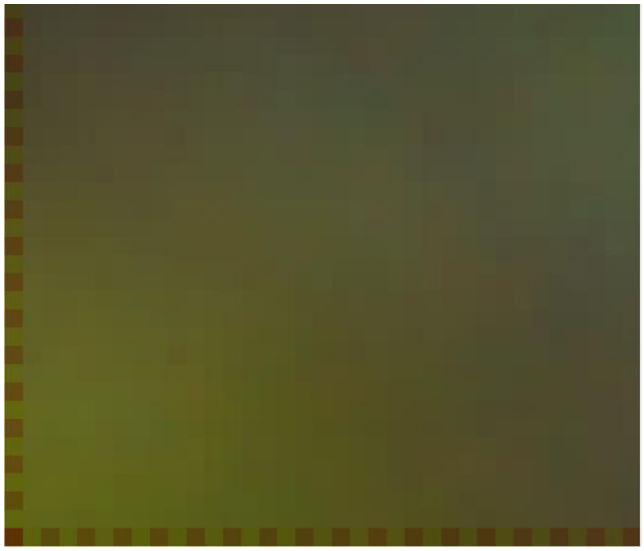
```
Warning: Image is too big to fit on screen;
displaying at 67%
```

1.3.2 - Συμπεράσματα από τα αποτελέσματα

Εκ πρώτης όψεως, το αποτέλεσμα φαίνεται να είναι πολύ ικανοποιητικό,

γεγονός που επαληθεύει την ορθότητα του αλγορίθμου. Με παραπάνω μελέτη, όμως, παρατηρούμε ότι δεν είναι όλα άψογα. Πιο συγκεκριμένα, παρατηρούμε την εμφάνιση ορισμένων artifacts.

Το πρώτο από αυτά είναι το border πάχους ενός pixel γύρω από την εικόνα που φαίνεται να μην ακολουθεί το demosaicing που γίνεται στα υπόλοιπα pixels, όπως φαίνεται στην εικόνα 6.



Εικόνα 6: Artifact border γύρω από την εικόνα

Αυτό είναι πιθανό να οφείλεται στην έλλειψη γειτονικών pixels ακριβώς πάνω στο όριο της εικόνας, άρα και στη μη ολοκληρωμένη συνέλιξη με τα kernels.

Δεύτερο artifact ή ατέλεια του αλγορίθμου είναι πως δε γίνεται αρκετά καλό demosaicing σε περιοχές απότομης εναλλαγής του χρώματος, όπως για παράδειγμα στα σύνορα του βλαστού. Αυτό φαίνεται στην εικόνα 7.

Εικόνα 7: Artifact λόγω απότομης εναλλαγής χρώματος

Η εμφάνιση του παραπάνω artifact είναι λογική, καθώς ο αλγόριθμος δουλεύει με βάση τη γειτονιά του pixel ανεξάρτητα με το ποιο είναι το pixel, δηλαδή χωρίς να έχει κάποιο αλγόριθμο για edge detection. Έτσι, τα συνοριακά pixel μιας επιφάνειας επηρεάζονται από τα pixel της άλλης επιφάνειας.

Όσο για το ερώτημα αν πετυχαίνουμε όντως averaging, ο αλγόριθμος αυτός με τα συγκεκριμένα kernels πετυχαίνει weighted averaging, δηλαδή δε διαιρούμε με τον αριθμό όλων των γειτονικών pixels, αλλά με τον αριθμό των γειτονικών pixels που έχουν χρώμα.

2 - Downsampling

2.1 - Στόχος της άσκησης

Ο στόχος της άσκησης είναι η μείωση των διαστάσεων μίας δοσμένης εικόνας σε επιθυμητές διαστάσεις. Συγκεκριμένα, η υλοποίηση της μείωσης διάστασης θα γίνει με 2 μεθόδους, τη μέθοδο nearest neighbor και τη μέθοδο bilinear interpolation. Οι μέθοδοι αυτοί έχουν σκοπό να 'συμπυκνώσουν' την πληροφορία πολλών pixels σε ένα μόνο pixel.

2.2 - Μέθοδοι Nearest Neighbor και Bilinear interpolation

Τόσο η μέθοδος nearest neighbor όσο και η μέθοδος bilinear interpolation για downsampling έχουν ως στόχο να συμπυκνώσουν την πληροφορία της εικόνας σε λιγότερα pixels αποδοτικά. Η ιδέα πίσω από τις μεθόδους αυτές, όμως, διαφέρει. Στην πραγματικότητα, η μέθοδος nearest neighbor διατηρεί αναλλοίωτη την πληροφορία που λαμβάνει από την αρχική εικόνα, αλλά ταυτόχρονα, χρησιμοποιεί και μικρό κομμάτι της αρχικής πληροφορίας για να παραγάγει το τελικό αποτέλεσμα. Αντίθετα, η μέθοδος bilinear interpolation δε διατηρεί την πληροφορία της αρχικής εικόνας αναλλοίωτη, αλλά παράγει νέα πληροφορία από την αρχική, όσο λαμβάνει, όμως, πληροφορία από μεγαλύτερο κομμάτι της εικόνας.

2.2.1 – Μέθοδος Nearest Neighbor

Η μέθοδος nearest neighbor για downsampling είναι αρκετά απλή. Βασίζεται στην ιδέα της ολικής διαγραφής της πληροφορίας. Αυτό σημαίνει, ότι η πληροφορία της αρχικής εικόνας που πλέον δε χρειάζεται να προβληθεί, χάνεται. Για παράδειγμα, εάν η εικόνα μας έχει μικρύνει κατά 4 φορές, τότε η μέθοδος κρατήσει μόνο 1 στα 4 pixel της αρχικής εικόνας, ενώ τα υπόλοιπα δε θα συμβάλουν καθόλου στο τελικό αποτέλεσμα.

Η τρόπος υλοποίησης της μεθόδου έχει ως εξής. Έστω η τριχρωματική εικόνα `xrgb` που αναπαρίσταται ως ένας NxMx3 πίνακας και έστω ότι N' και M' είναι οι νέες διαστάσεις που θέλουμε να έχει η εικόνα, όπου N'<N και M'<M. Αρχικά, υπολογίζουμε το συντελεστή σμίκρυνσης σε κάθε διάσταση, δηλαδή τους αριθμούς l = N/N' και w = M/M'. Στη συνέχεια, κάθε pixel της *εικόνας εξόδου* `xrgbres`, έστω (x,y), το αντιστοιχίζουμε με το pixel (round(l*x), round(w*y)) της *εικόνας εισόδου* `xrgb`. Τέλος παίρνουμε το χρώμα του pixel (round(l*x), round(w*y)) της εικόνας εισόδου και το αποδίδουμε στο pixel (x,y) της εικόνας εξόδου. Με τον τρόπο αυτό, έχουμε δημιουργήσει μια νέα εικόνα μικρότερης διάστασης, που έχει διατηρήσει ένα ποσοστό της αρχικής εικόνας αυτούσιο, ενώ το υπόλοιπο έχει χαθεί. Για την παραπάνω διαδικασία στα πλαίσια της εργασίας, έχει υλοποιηθεί η συνάρτηση `function xrgbres = myresize(xrgb , N, M, method)`, για `method = 'nearest'`.

2.2.2 – Μέθοδος Bilinear interpolation

Η μέθοδος bilinear interpolation για downsampling είναι πιο πολύπλοκη από τη nearest neighbor. Η βασική διαφορά της από άποψη ιδεολογίας είναι η διατήρηση μεγαλύτερου μέρους της πληροφορίας της αρχικής εικόνας, όσο αυτό είναι εφικτό. Αυτό σημαίνει ότι τα pixels της εικόνας εξόδου είναι ίσως πιο αντιπροσωπευτικά της περιοχής της εικόνας εισόδου από την οποία προέρχονται, αλλά δε διατηρούν πλέον αυτούσια την πληροφορία της αρχικής εικόνας.

Ο τρόπος υλοποίησης της μεθόδου βασίζεται στη γραμμική παρεμβολή σε 2 διαστάσεις. Έστω η τριχρωματική εικόνα `xrgb` που αναπαρίσταται ως ένας NxMx3 πίνακας και έστω ότι N' και M' είναι οι νέες διαστάσεις που θέλουμε να έχει η εικόνα, όπου N'<N και M'<M. Αρχικά, υπολογίζουμε το συντελεστή σμίκρυνσης σε κάθε διάσταση, δηλαδή τους αριθμούς l = N/N' και w = M/M'. Στη συνέχεια, κάθε pixel της *εικόνας εξόδου* `xrgbres`, έστω (x,y), το αντιστοιχίζουμε σε ένα ορθογώνιο παραλληλόγραμμο της *εικόνας εισόδου* `xrgb`, το οποίο περιγράφεται από 4 pixels (x1, y1), (x1,y2), (x2,y1), (x2, y2). Τα x1, x2, y1, y2 υπολογίζονται ως εξής:

- x1 = round(l*x)
- x2 = round(x1 - l)
- y2 = round(w*y)
- y1 = round(y1 - w)

Μετά, υπολογίζουμε το ‘κέντρο’ του χωρίου αυτού (x’,y’) ως x’ = (x1+x2)/2 και y’ = (y1+y2)/2. Εφαρμόζουμε τον τύπο του bilinear interpolation (εικόνα 8) μεταξύ των κορυφών του χωρίου και βρίσκουμε το χρώμα που προκύπτει.

$$
\begin{aligned}
f(x,y) &\approx \frac{y_2-y}{y_2-y_1} f(x,y_1) + \frac{y-y_1}{y_2-y_1} f(x,y_2) \\
&= \frac{y_2-y}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{11}) + \frac{x-x_1}{x_2-x_1} f(Q_{21})\right) + \frac{y-y_1}{y_2-y_1}\left(\frac{x_2-x}{x_2-x_1} f(Q_{12}) + \frac{x-x_1}{x_2-x_1} f(Q_{22})\right) \\
&= \frac{1}{(x_2-x_1)(y_2-y_1)} \left(f(Q_{11})(x_2-x)(y_2-y) + f(Q_{21})(x-x_1)(y_2-y) + f(Q_{12})(x_2-x)(y-y_1) + f(Q_{22})(x-x_1)(y-y_1)\right) \\
&= \frac{1}{(x_2-x_1)(y_2-y_1)} \begin{bmatrix} x_2-x & x-x_1 \end{bmatrix} \begin{bmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{bmatrix} \begin{bmatrix} y_2-y \\ y-y_1 \end{bmatrix}.
\end{aligned}
$$

Εικόνα 8: Τύπος bilinear interpolation, πηγή Wikipedia

Τέλος, αποδίδουμε το χρώμα αυτό στο pixel (x,y) της εικόνας εξόδου `xrgbres`. Με τον τρόπο αυτό, στην πραγματικότητα έχουμε κάνει μια ‘περίληψη’ των χρωμάτων των κορυφών του χωρίου σε ένα μόνο pixel, άρα έχουμε χρησιμοποιήσει μεγαλύτερο μέρος της πληροφορίας (σε σχέση με το nearest neighbor) για να δημιουργήσουμε ένα νέο χρώμα, αλλά δε διατηρούμε τα αρχικά χρώματα της φωτογραφίας. Για την παραπάνω διαδικασία στα πλαίσια της εργασίας, έχει υλοποιηθεί η συνάρτηση `function xrgbres = myresize(xrgb , N, M, method)`, για `method = ‘linear’`.

2.3 - Αποτελέσματα και συμπεράσματα

Μετά την εφαρμογή της διαδικασίας demosaicing πάνω στα δεδομένα `march.mat` όπως περιεγράφηκε στο κεφάλαιο 1, με σκοπό να πάρουμε μια τριχρωματική εικόνα, εφαρμόσθηκαν οι 2 παραπάνω τεχνικές downsampling στην εικόνα αυτή μέσω κώδικα Matlab και πήραμε τα παρακάτω αποτελέσματα.

2.3.1 - Αποτελέσματα

Στο πρώτο πείραμα, αλλάξαμε το μέγεθος της εικόνας από 960x1280 σε 240x320 με τη μέθοδο nearest neighbor και το αποτέλεσμα είναι αυτό που φαίνεται στις εικόνα 9.

Εικόνα 9: Downsampled εικόνα με nearest neighbor

Στο δεύτερο πείραμα, αλλάξαμε το μέγεθος της εικόνας από 960x1280 σε 200x300 με τη μέθοδο bilinear interpolation και το αποτέλεσμα είναι αυτό που φαίνεται στις εικόνα 10.

Εικόνα 10: Downsampled εικόνα με bilinear interpolation

2.3.2 - Συμπεράσματα από τα αποτελέσματα

Τα αποτελέσματα της σμίκρυνσης φαίνονται να είναι ικανοποιητικά. Αυτό επιβεβαιώνει την ορθότητα των αλγορίθμων που χρησιμοποιήσαμε. Με λίγη προσοχή, όμως, παρατηρούμε ότι υπάρχουν διαφορές στο αποτέλεσμα των αλγορίθμων πάνω στην εικόνα και προβλήματα από το demosaicing που παραμένουν ακόμα και μετά τη σμίκρυνση.

Αρχικά, το artifact border και το artifact στα όρια των επιφανειών της αρχικής εικόνας φαίνεται να διατηρείται στις νέες, μικρότερες εικόνες. Αυτό είναι

λογικό και καθησυχαστικό, διότι μας διαβεβαιώνει ότι η σμίκρυνση όντως αντιπροσωπεύει την αρχική εικόνα και τις ιδιότητές της.

Στη συνέχεια, με μεγέθυνση της εικόνας 9 (εικόνα 11) μπορούμε να δούμε ότι η εικόνα έχει διατηρήσει τις ακμές της αρχικής εικόνας και γενικά, ενώ τα γειτονικά pixels αλλάζουν αρκετά απότομα χρώμα, η εικόνα είναι ευδιάκριτη και ‘καθαρή’. Βέβαια, ενώ η αρχική εικόνα δεν έχει αρκετά λεπτές ακμές για να το παρουσιάσουμε, με τη μέθοδο nearest neighbor διατρέχουμε τον κίνδυνο να διαγραφούν πολύ λεπτές ακμές (πάχους μερικών pixels, ίσως), λόγω της διαγραφής πληροφορίας.

Εικόνα 11: Μεγέθυνση της εικόνας 9

Αντίστοιχα, με μεγέθυνση της εικόνας 10 (εικόνα 12), μπορούμε να δούμε ότι η μέθοδος bilinear interpolation εισάγει ένα ‘θόλωμα’ στην εικόνα, δηλαδή οι ακμές της παύουν να είναι τόσο ευδιάκριτες.

Εικόνα 12: Μεγέθυνση της εικόνας 10

3 - Quantization

3.1 - Στόχος της άσκησης

Ο στόχος της άσκησης αυτής είναι ο κβαντισμός και ο αποκβαντισμός μιας τιμής και κατ' επέκταση των τιμών φωτεινότητας των χρωμάτων μίας εικόνας. Αυτό, στην ουσία, σημαίνει να βρούμε ένα τρόπο με τον οποίο να μπορούμε να πάμε από ένα συνεχές πεδίο τιμών σε ένα διακριτό. Αυτό είναι απαραίτητο να γίνει για κάθε εικόνα (και οτιδήποτε δεν έχει πεπερασμένο μήκος/ακρίβεια και αποφασίζουμε να αποθηκεύσουμε σε έναν υπολογιστή), διότι ενώ μία τιμή φωτεινότητας μπορεί να έχει άπειρη ακρίβεια, η μνήμη ενός υπολογιστή είναι πεπερασμένη. Συνεπώς, πρέπει να καθορίσουμε την ακρίβεια με την οποία θα αναπαρασταθούν τα χρώματα της εικόνας μας, ανάλογα με τους πόρους μνήμης που είναι διαθέσιμοι στο σύστημά μας.

3.2 - Κβαντιστής και αποκβαντιστής τιμών

Για τη διαδικασία του κβαντισμού μιας τιμής, θα δημιουργήσουμε ένα απλό πρόγραμμα που εκτελεί τη διαδικασία αντιστοίχισης από το σύνολο των πραγματικών R στο σύνολο των ακεραίων Z. Με αντίστοιχη λογική, θα φτιάξουμε έναν αποκβαντιστή, που θα εκτελεί την αντίστροφη διαδικασία.

3.2.1 - Κβαντιστής τιμών

Ο κβαντιστής τιμών που θα δημιουργήσουμε είναι ομοιόμορφος και συμμετρικός κβαντιστής και είναι μια συνάρτηση στο Matlab με όνομα `function q = myquant(x, w)`. Γενικά, θέλουμε να μπορούμε να κβαντίσουμε οποιαδήποτε τιμή στο R. Όμως, ιδιαίτερο ενδιαφέρον παρουσιάζουν οι τιμές στο [0,1], διότι εκεί βρίσκονται και οι τιμές φωτεινότητας των χρωμάτων της εικόνας. Το q είναι η κβαντισμένη τιμή που επιστρέφει η συνάρτηση, το x η τιμή προς κβάντιση και w το πλάτος ζώνης κβαντισμού, δηλαδή το εύρος των τιμών που θα αντιστοιχηθούν σε

μία στάθμη κβαντισμού. Κατά τη δημιουργία του κβαντιστή κάνουμε κάποιες συμβάσεις:

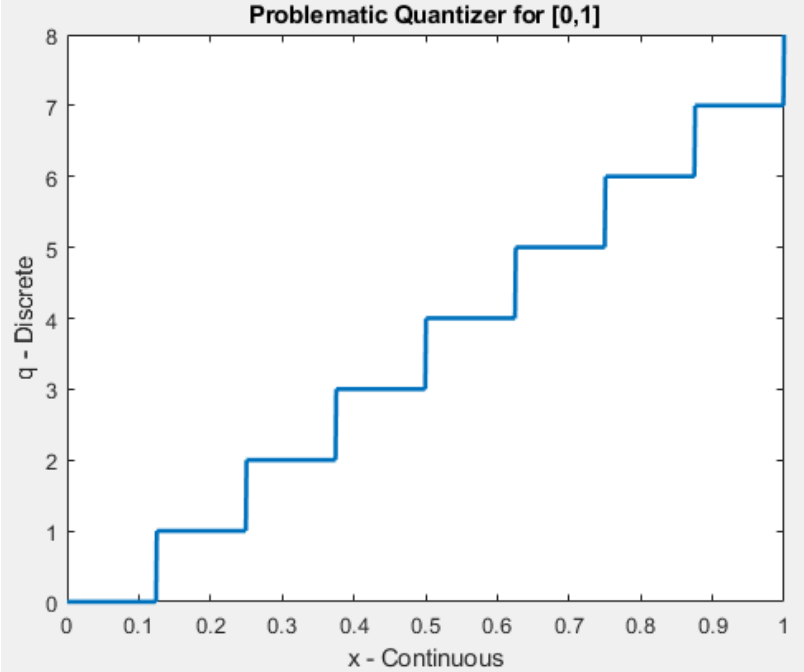
- Οι στάθμες κβαντισμού είναι αριθμοί στο σύνολο Z και επίσης [0,∞) για x ≥0 και (- ∞ , -1] για x<0
- Το σύνολο [n*w, (n+1)*w] πρέπει κβαντίζεται στη στάθμη κβαντισμού n.

Η τελευταία σύμβαση παρουσιάζει κάποια προβλήματα. Δεν είναι δυνατό στις στάθμες κβαντισμού να αντιστοιχούν σύνολα του τύπου [n*w, (n+1)*w] , καθώς η τιμή (n+1)*w θα άνηκε και στάθμη n αλλά και στην n+1. Από την άλλη πλευρά, για τη συγκεκριμένη εργασία, θα μας βόλευε η οριακή τιμή 0 να κβαντίζεται στη στάθμη 0, αλλά η οριακή τιμή 1 να κβαντίζεται στη στάθμη που ανήκει και η τιμή 1-w, ώστε να μην έχουμε μία ακόμα στάθμη κβαντισμού. Αυτό γίνεται, διότι οι στάθμες κβαντισμού ορίζονται από τη διαθέσιμη μνήμη μας, δηλαδή πόσα bits έχουμε. Οι στάθμες είναι 2^bits, άρα αν, για παράδειγμα, είχαμε 3 bits θα είχαμε 8 στάθμες, αλλά η τιμή 1 θα έμπαινε σε μια $9^{η}$ στάθμη (όπως φαίνεται στην εικόνα 13), δημιουργώντας, ίσως, κάποιο σφάλμα. Για το λόγο αυτό, αποφασίζουμε να κάνουμε μία ακόμα σύμβαση:

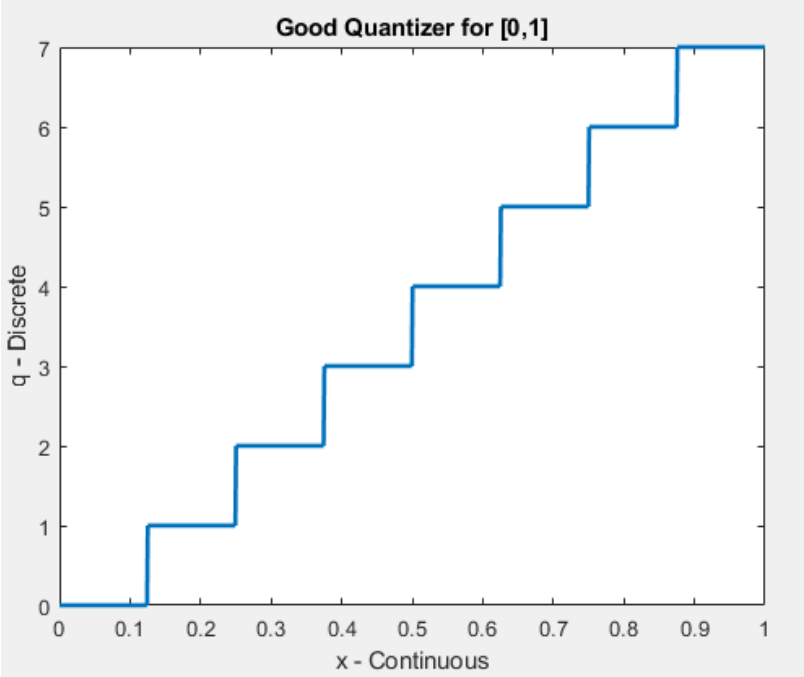
- Πριν την κβάντιση, αφαιρούμε από την αρχική τιμή x μια πολύ μικρή ποσότητά της, την τιμή x*(1/10000)

Το παραπάνω εξασφαλίζει ότι το 0 θα μπει στη στάθμη 0, αφού 0 – 0*(1/10000) = 0, ενώ η τιμή 1 δε θα δημιουργήσει μια νέα στάθμη, αφού 1-1*(1/10000) = 0.99999 που είναι πολύ κοντά στο 1.

Αφού, λοιπόν, έχουμε κάνει την παραπάνω αφαίρεση, η στάθμη κάθε αριθμού είναι ο αμέσως μικρότερος ακέραιος (floor) του αριθμού που προκύπτει από την πράξη x/w. Ο κβαντιστής αυτός φαίνεται στην εικόνα 14.



Εικόνα 13: Προβληματικός κβαντιστής, bits = 3 , x = [0,1]



Εικόνα 14: Καλός κβαντιστής, bits=3, x = [0,1]

Σημειώνεται ότι ο παραπάνω κβαντιστής έχει υλοποιηθεί κατάλληλα, ώστε να λειτουργεί με τιμές, διανύσματα και δισδιάστατους πίνακες.

3.2.2 - Αποκβαντιστής τιμών

Ο αποκβαντιστής που δημιουργήσαμε έχει μόνο μια σύμβαση, ότι θα αποκβαντίζει τις στάθμες στο μέσο της ζώνης κβαντισμού. Η συνάρτηση Matlab που υλοποιεί τη διαδικασία αυτή είναι η `function x = mydequant(q, w).` Η συνάρτηση αντιστοιχίζει τη στάθμη q στην τιμή (q+(q+1))*w/2. Για να γίνει σωστά αυτή η διαδικασία, αν μια στάθμη είναι αρνητική, πρέπει να την αυξήσουμε κατά 1.

Σημειώνεται ότι ο παραπάνω αποκβαντιστής έχει υλοποιηθεί κατάλληλα, ώστε να λειτουργεί με τιμές, διανύσματα και δισδιάστατους πίνακες.

3.2.3 - Κβαντιστής και αποκβαντιστής εικόνας

Έχοντας υλοποιήσει τον κβαντιστή και τον αποκβαντιστή για τιμές, μπορούμε εύκολα να δημιουργήσουμε τον κβαντιστή και αποκβαντιστή για τις τιμές φωτεινότητας των χρωμάτων μιας εικόνας. Η διαδικασία του κβαντισμού υλοποιείται από τη συνάρτηση `function q = imagequant(x, w1 , w2 , w3).` Η λειτουργία της είναι αρκετά απλή. Απλά χωρίζουμε την εικόνα x σε δισδιάστατους πίνακες των χρωμάτων R, G και B και στη συνέχεια εφαρμόζουμε τη συνάρτηση myquant για w1, w2 και w3. Τέλος, ανασυνθέτουμε τον τρισδιάστατο πίνακα της εικόνας q, που περιέχει τις κβαντισμένες τιμές. Αντίστοιχη διαδικασία κάνουμε για τον αποκβαντισμό, με τη συνάρτηση `function x = imagedequant(q, w1 , w2 , w3).`

3.3 - Αποτελέσματα και συμπεράσματα

3.3.1 - Αποτελέσματα

Αφού έχουμε υλοποιήσει τις παραπάνω συναρτήσεις, μπορούμε να δούμε, στην εικόνα 15, πως θα ήταν η αρχική μας εικόνα, εάν είχαμε μόνο 3 bits/χρώμα, δηλαδή αν κάθε χρώμα έπαιρνε μόνο 2^3 = 8 διακριτές τιμές.

Εικόνα 15: Εικόνα με 3 bits/χρώμα

3.3.2 - Συμπεράσματα από τα αποτελέσματα

Το αποτέλεσμα του κβαντισμού και αποκβαντισμού της εικόνας είναι ικανοποιητικό, γεγονός που επιβεβαιώνει την ορθότητα των αλγορίθμων που σχεδιάσαμε.

Τα artifacts που παρατηρήσαμε σε άλλα στάδια παραμένουν. Αυτό, όμως, είναι καθησυχαστικό, διότι δε θα περιμέναμε να φύγουν.

Μια ακόμα παρατήρηση είναι ότι η ανάλυση της εικόνας πέφτει και η εικόνα δεν είναι πλέον όσο ομαλή ήταν πριν. Αυτό είναι αναμενόμενο, αφού μειώσαμε πολύ την ακρίβεια με την οποία αναπαρίστανται τα χρώματα.

4 - PPM representation

4.1 - Στόχος της άσκησης

Ο στόχος της άσκησης αυτή είναι η αποθήκευση της εικόνας ως ένα αρχείο με ένα συγκεκριμένο πρότυπο, το πρότυπο ppm, ώστε να είναι αναγνώσιμη εικόνα για έναν υπολογιστή.

4.2 - Το πρότυπο PPM

Το πρότυπο ppm είναι ένα απλό πρότυπο για την αποθήκευση εικόνων σε κβαντισμένη μορφή. Αποτελείται από ένα header και τα δεδομένα της εικόνας, δηλαδή το χρώμα κάθε pixel. Το header έχει το magic number 'P6' (για τη δική μας περίπτωση) και ακολουθείται από τις διαστάσεις της εικόνας και τον αριθμό των σταθμών κβάντισης της εικόνας. Ένα παράδειγμα είναι 'P6 1280 960 8 '. Σημειώνεται ότι τα κενά παίζουν ρόλο για το πρότυπο. Μετά ακολουθούν τα δεδομένα των pixel. Παίρνουμε τα pixel κάθε σειράς (από πάνω προς τα κάτω) διαδοχικά (από αριστερά προς τα δεξιά) και τα γράφουμε στο αρχείο μας σε binary μορφή στο αρχείο μας. Προφανώς, χρειάζεται προσοχή στον αριθμό bytes που χρησιμοποιούμε για την αποθήκευση των κβαντισμένων τιμών. Κάθε pixel το γράφουμε ως τρία διαδοχικά στοιχεία r/g/b. Τέλος, αποθηκεύουμε το αρχείο με κατάληξη '.ppm' . Την παραπάνω διαδικασία υλοποιεί η συνάρτηση Matlab `function saveasppm(x, filename , K)` της εργασίας.

4.3 - Αποτελέσματα και συμπεράσματα

4.3.1 - Αποτελέσματα

Στα πλαίσια της εργασίας, υλοποιήθηκε το εξής pipeline:

- Μετατροπή της εικόνας από φίλτρο Bayer σε πραγματική εικόνα rgb

- Downsampling της rgb εικόνας σε διαστάσεις 150x200 με τη μέθοδο bilinear interpolation.
- Κβαντισμός της εικόνας με 3 bits/χρώμα.
- Αποθήκευση της εικόνας ως αρχείο ppm

Η εικόνα που παίρνουμε μετά την εκτέλεση του παραπάνω pipeline φαίνεται στην εικόνα 16.

Εικόνα 16: Εικόνα μετά το pipeline

Σε μεγέθυνση, η παραπάνω εικόνα φαίνεται στην εικόνα 17.

Εικόνα 17: Μεγεθυμένη εικόνα μετά το pipeline

4.3.2 - Συμπεράσματα από τα αποτελέσματα

Το αποτέλεσμα του pipeline είναι ικανοποιητικό, γεγονός που επιβεβαιώνει την ορθότητα των αλγορίθμων που σχεδιάσαμε. Είναι εμφανές ότι η αποθήκευση ως αρχείο ppm αναγνωρίζεται από τον υπολογιστή, καθώς το Matlab είναι ικανό να διαβάσει και να εμφανίσει την εικόνα από το αρχείο ppm. Οι υπόλοιπες παρατηρήσεις έχουν να κάνουν με προηγούμενα στάδια της εργασίας και έχουν σχολιαστεί στο αντίστοιχο κεφάλαιο.