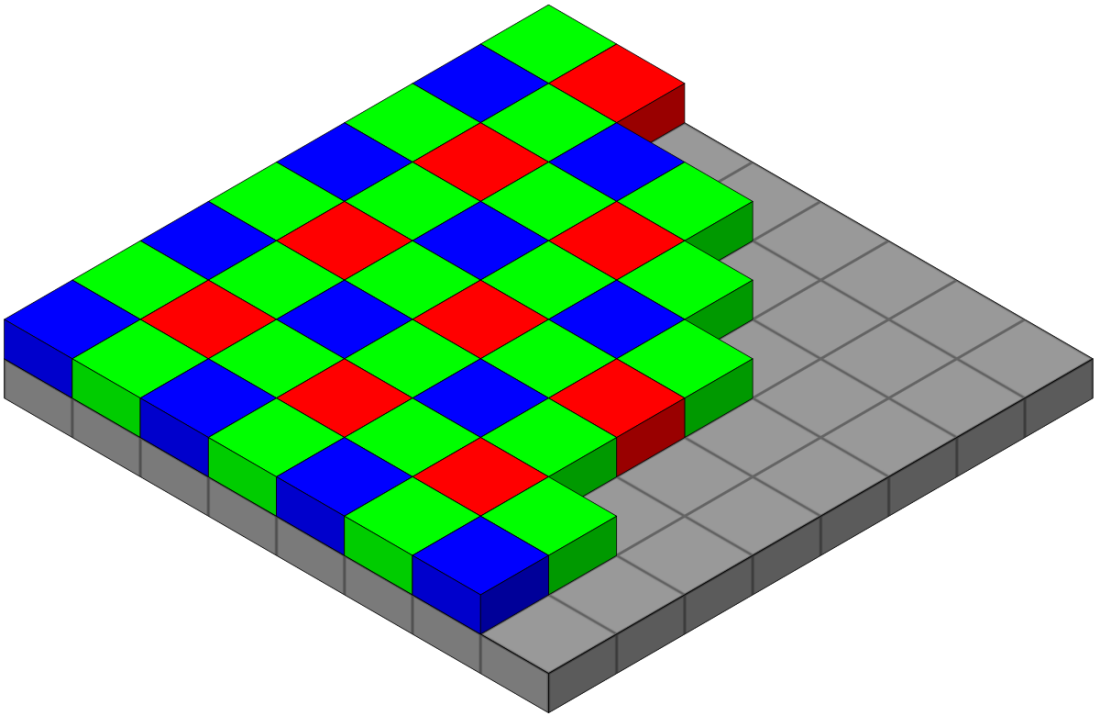
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ψηφιακή Επεξεργασία Εικόνας

Εργασία 1η – Εαρινό Εξάμηνο 2020/2021



Καβελίδης Φραντζής Δημήτριος – ΑΕΜ 9351

18/4/2021

Θέμα:

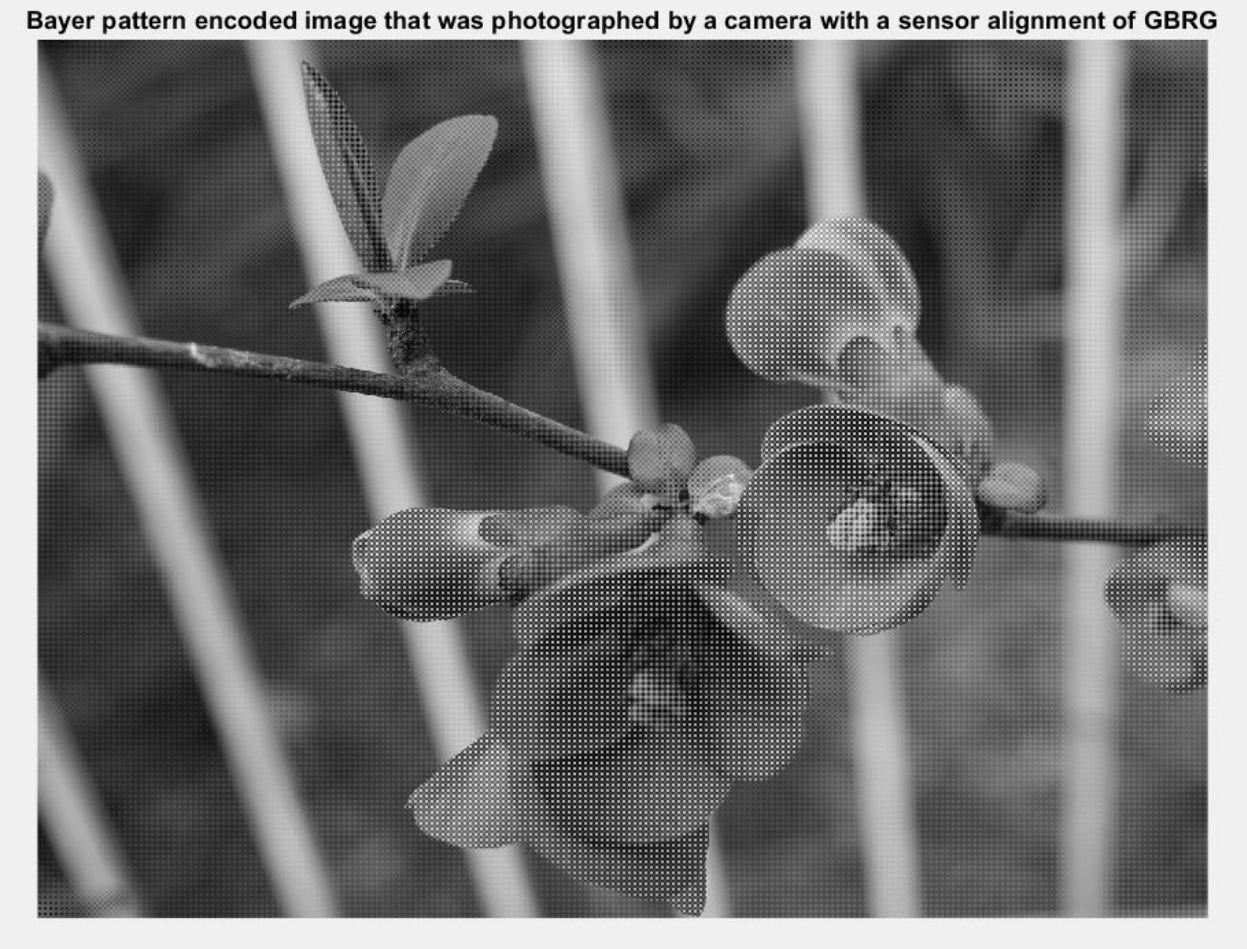
Η πρώτη εργασία του μαθήματος αφορά την υλοποίηση των παρακάτω:

1. Ανακατασκευή τριχρωματικής εικόνας RGB από εικόνα Bayer
2. Υποδειγματοληψία της RGB εικόνας
3. Κβαντισμός των τιμών φωτεινότητας ανά κανάλι
4. Αποθήκευση των κβαντισμένων τιμών σε δυαδική μορφή.

Για τους σκοπούς της εργασίας θεωρείστε ότι έχετε τις πραγματικές τιμές φωτεινότητας στο διάστημα [0, 1] τις οποίες κατέγραψε η φωτογραφική μηχανή. Οι τιμές φωτεινότητας που κατέγραψε η φωτογραφική μηχανή σας δίνονται στο αρχείο march.mat.

Ανάλυση:

Αρχικά, εμφανίζουμε την Bayer Pattern encoded εικόνα που προκύπτει από τις τιμές φωτεινότητας της φωτογραφικής μηχανής:



Εικόνα 1 – Εικόνα Bayer από τις τιμές φωτεινότητας του αισθητήρα της φωτογραφικής μηχανής.

1. **<u>Φίλτρο Bayer:</u>**

Στο πρώτο μέρος της εργασίας, καλούμαστε να φτιάξουμε μία συνάρτηση (**bayer2rgb.m**) που ανακατασκευάζει την εικόνα μας σε RGB από Bayer γεμίζοντας τα κενά pixel κάθε καναλιού της εικόνας με τον μέσο όρο των γειτονικών τους. Για να το πετύχουμε αυτό, θα χρησιμοποιήσουμε τα kernels που είναι ευρέως γνωστά κατά την διγραμμική παρεμβολή (bilinear interpolation). Τα kernels αυτά είναι τα γνωστά Gaussian kernel, και αυτό που κάνουν στην ουσία είναι ότι δίνουν μεγαλύτερη βαρύτητα στο κεντρικό pixel της μάσκας και μικρότερη βαρύτητα όσο απομακρυνόμαστε από αυτό, δηλαδή μας δίνει έναν μέσο όρο με συγκεκριμένα βάρη.
Τα kernels αναλυτικά:

$$\boldsymbol{kernelR = kernelB = \frac{1}{4} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}}$$

$$\boldsymbol{kernelG = \frac{1}{4} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}}$$

Η φωτογραφία που προκύπτει από την ανακατασκευή που κάναμε φαίνεται όταν τρέχουμε το αρχείο demo1.m :

Εικόνα 2 – Ανακατασκευασμένη RGB εικόνα.

Αξίζει να σημειωθεί, ότι επειδή χρησιμοποιούμε την συγκεκριμένη μέθοδο σε συνδυασμό με συνέλιξη, όπως είναι λογικό, περιμένουμε να έχουμε artifacts στην περίμετρο, δηλαδή όχι «σωστά» averaged pixels. Έτσι, για να αντιμετωπίσουμε αυτήν την πάθηση, θα χρησιμοποιήσουμε μία μέθοδο που να ομαλοποιεί αυτό το φαινόμενο. Αυτό που θα κάνουμε είναι ότι ουσιαστικά θα αντιγράψουμε στην εξωτερική περίμετρο την περίμετρο που βρίσκεται δύο επίπεδα πιο εσωτερικά. Με αυτόν τον τρόπο είμαστε συνεπείς ως προς το Bayer Pattern της εικόνας και προκύπτουν ικανοποιητικά αποτελέσματα. Έτσι τελικά προκύπτει η εικόνα που έχουμε παραπάνω.

Ο τρόπος που λειτουργεί ο κώδικας στη συνάρτηση **bayer2rgb** είναι ότι φτιάχνουμε 3 πίνακες για τα χρωματικά μας κανάλια μοιράζοντάς τους τις τιμές της αρχικής μας εικόνας με βάση το alignment των αισθητήρων της φωτογραφικής μας μηχανής, δηλαδή το Bayer Pattern που χρησιμοποιεί. Στην προκειμένη έχουμε GBRG format. Στην συνέχεια απλώς χρησιμοποιούμε συνέλιξη με κάποιο kernel ώστε να πετύχουμε averaging, το οποίο πετυχαίνουμε διαλέγοντας τα κατάλληλα kernel που αναφέραμε παραπάνω. Τέλος, αντιμετωπίζουμε τα φαινόμενα στην περίμετρο.

2. **<u>Υποδειγματοληψία εικόνας:</u>**

   Σε αυτήν την ενότητα παρουσιάζονται τα αποτελέσματα που προκύπτουν κατά την υποδειγματοληψία (downsampling) της εικόνας, χρησιμοποιώντας 2 διαφορετικές μεθόδους. Τα αποτελέσματα παρουσιάζονται παρακάτω.
   Με τη μέθοδο του κοντινότερου γείτονα και μέγεθος 240*320:

Εικόνα 3 – Μείωση μεγέθους/Υποδειγματοληψία χρησιμοποιώντας τη μέθοδο του κοντινότερου γείτονα (Nearest Neighbor Interpolation).

Αντίστοιχα με τη μέθοδο της διγραμμικής παρεμβολής και μέγεθος 200*300:

Εικόνα 4 - Μείωση μεγέθους/Υποδειγματοληψία χρησιμοποιώντας τη μέθοδο του διγραμμικής παρεμβολής (Bilinear Interpolation).

Γενικά, η μέθοδος της διγραμμικής παρεμβολής φαίνεται να κάνει ένα ελαφρύ smoothing/blur στην εικόνα, επομένως «αντιστέκεται» εν μέρει στον τρόπο που (αναπόφευκτα) χάνεται η πληροφορίας λόγω της υποδειγματοληψίας. Παρόλα αυτά, σε αυτή τη μέθοδο ενδέχεται να έχουμε νέες τιμές σε ορισμένα pixel. Η μέθοδος του κοντινότερου γείτονα απ' την άλλη, παρόλο που είναι γρηγορότερη, δίνει μία πιο blocky αίσθηση στην εικόνα λόγω των σφαλμάτων της.

Ο τρόπος που λειτουργεί ο κώδικας στη συνάρτηση myresize είναι ο εξής:
α) Αν η μέθοδος είναι η μέθοδος κοντινότερων γειτόνων, πρώτα φτιάχνουμε τον λόγο της ζητούμενης διάστασης προς την πραγματική διάσταση της εικόνας για την κάθε μία από τις 2 διαστάσεις. Έπειτα, υπολογίζουμε τις θέσεις (index) παρεμβολής και τις χρησιμοποιούμε για να ξαναφτιάξουμε την εικόνα μας (μέσω των 3 καναλιών) χρησιμοποιώντας όμως τώρα μόνο τα index που υπολογίσαμε. Κάνουμε την παραπάνω διαδικασία επιλέγοντας πρώτα γραμμές και μετά στήλες για το κάθε χρώμα. Έτσι, έχουμε πάρει συγκεκριμένα pixels από την αρχική εικόνα για την ανακατασκευή της μικρότερης εικόνας.
β) Αν η μέθοδος είναι η μέθοδος διγραμμικής παρεμβολής, πρώτα φτιάχνουμε τον λόγο της ζητούμενης διάστασης προς την πραγματική διάσταση της εικόνας για την κάθε μία από τις 2 διαστάσεις. Έπειτα, ακολουθούμε τον παρακάτω τύπο υλοποίησης του αλγόριθμου διγραμμικής

παρεμβολής:

$$f(x,y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy,$$

και το κάνουμε αυτό για κάθε 4άδα (τετραγωνάκι) από pixel θεωρώντας κάθε φορά ότι το πρώτο (πάνω αριστερά) είναι το **f(0,0)**, δημιουργώντας έτσι ένα νέο pixel και καταλήγοντας τελικά σε μία εικόνα με νέες διαστάσεις. Στις ακραίες περιπτώσεις απλώς αντιγράφεται η τελευταία από την προτελευταία σειρά/στήλη.

3. **Κβαντισμός και αποκβαντισμός:**

Στην συνέχεια, υλοποιούμε ένα σύστημα συναρτήσεων κβαντισμού και απόκβαντισμού της τριχρωματικής εικόνας. Με βάση τον αριθμό των bits (n) που χρησιμοποιούμε ανά χρώμα, κβαντίζουμε τις τιμές των pixel σε αυτές τις σταθερές στάθμες, δηλαδή τα 2^n επίπεδα. Συγκεκριμένα, χρησιμοποιούμε τον κβαντιστή – αποκβαντιστή mid – riser uniform (συμμετρικός ομοιόμορφος):

$$\boldsymbol{Q(x) = w \cdot (floor(x/w) + 1/2)}$$

Ο κβαντιστής είναι:

$$\boldsymbol{q = floor(x/w)}$$

ενώ ο αποκβαντιστής είναι:

$$\boldsymbol{y = w \cdot (q + 1/2)}$$

όπου $\boldsymbol{w}$ το πλάτος ζώνης κβαντισμού:

$$\boldsymbol{w = \frac{x_{max} - x_{min}}{2^n}}$$

Στη συγκεκριμένη ενότητα, λειτουργία του κώδικα είναι τετριμμένη.

Η κβαντισμένη-αποκβαντισμένη εικόνα που προκύπτει:

Εικόνα 5 - Κβαντισμένη – Αποκβαντισμένη εικόνα.

4. **<u>Πρότυπο PPM:</u>**

Στο τελευταίο αυτό μέρος, υλοποιούμε έναν κωδικοποιητή (**saveasppm.m**) σύμφωνα με το πρότυπο **PPM**.

Το συγκεκριμένο πρότυπο λειτουργεί ως εξής:

α) Στο πρώτο μέρος τέτοιων αρχείων υπάρχουν τα bytes που αν αναγνωσθούν ως **ASCII** χαρακτήρες, αποτελούν 4 παραμέτρους ακολουθούμενες από έναν χαρακτήρα κενού ή αλλαγή γραμμής. Οι παράμετροι είναι η σταθερή "λέξη" P6, ο αριθμός pixels για το πλάτος της εικόνας, ο αριθμός pixels για το ύψος της εικόνας, και ο αριθμός των στάθμεων κβαντισμού K.

β) Στο υπόλοιπο αρχείο περιέχονται οι αντιστοιχισμένες ακέραιες τιμές (των συμβόλων κβαντισμού) των τιμών φωτεινότητας του κάθε pixel. Η αποθήκευση των τιμών των pixel γίνονται σε τριάδες (διαδοχικά με σειρά κόκκινο -> πράσινο -> μπλε) ανά γραμμή.

Στο αρχείο **demo4.m**, γίνεται μία χρήση όλων των συναρτήσεων που κληθήκαμε να φτιάξουμε κατά τη διάρκεια της εργασίας, ώστε να αποδείξουμε την αποτελεσματικότητα/χρηστικότητα της τελευταίας μας συνάρτησης (**saveassppm.m**). Έτσι, αφού φτιάξουμε την εικόνα, την

υποδειγματοληπτίσουμε και την κβαντίσουμε, αποθηκεύουμε την εικόνα μέσω της συνάρτησης και στη συνέχεια εμφανίζουμε την αποθηκευμένη κατά το πρότυπο PPM εικόνα:

Εικόνα 6 – Αποθηκευμένη κατά ppm, κβαντισμένη, resized εικόνα.

Στην συνάρτηση saveasppm λειτουργούμε ως εξής:
Ανοίγουμε ένα αρχείο με την **fopen()**, γράφουμε με την fprintf στην πρώτη σειρά του το header με τις απαραίτητες πληροφορίες του αρχείου και στη συνέχεια κάνουμε έλεγχο για τις στάθμες . Αν αυτές είναι περισσότερες από 255, τότε χρειαζόμαστε 2 bytes, επομένως χρησιμοποιούμε την εντολή **fwrite()**, με τα κατάλληλα ορίσματα ('uint16', 'ieee-be') για να γράψουμε στο αρχείο την τιμή του pixel για το κάθε χρώμα. Αυτή η εντολή ουσιαστικά είναι 3 φορές (1 για κάθε χρώμα) μέσα σε 2 for για να διατρέξουν όλη την εικόνα. Αντίστοιχα, οι στάθμες είναι λιγότερες από 255, χρειαζόμαστε μόνο 1 byte, επομένως θα κάνουμε ακριβώς το ίδιο, μόνο με διαφορετικό όρισμα στην **fwrite()**, το 'uint8'.
Τέλος, το πρόγραμμα demo4.m χρησιμοποιεί την εικόνα που βρίσκεται στο δοσμένο αρχείο, κάνει υποδειγματοληψία, την κβαντίζει και έπειτα αποθηκεύει την εικόνα σε αρχείο ppm. Ουσιαστικά, ο τρόπος που λειτουργεί το συγκεκριμένο πρότυπο είναι σαν αποκβαντιστής, κάτι που φαίνεται στην εικόνα 6.

Σημείωση: Αν σε αυτό το βήμα κάνουμε την ίδια διαδικασία χωρίς να κάνουμε resize, θα συνειδητοποιήσουμε ότι οι εικόνα που θα προκύψει δεν είναι ίδια με αυτήν που προκύπτει από το demo3.m (όπως δηλαδή θα περιμέναμε), αλλά έχει μία ανεπαίσθητη διαφορά (τα χρώματα φαίνονται ελάχιστα πιο σκούρα). Αυτό έχει να κάνει με τον τρόπο με τον οποίο διαβάζει το ppm αρχείο η συνάρτηση imshow και κάνει τον αποκβαντισμό του, ο οποίος ενδεχομένως να διαφέρει από τον τρόπο που το χειρίζεται το δικό μας σύστημα κβαντιστή – αποκβαντιστή.