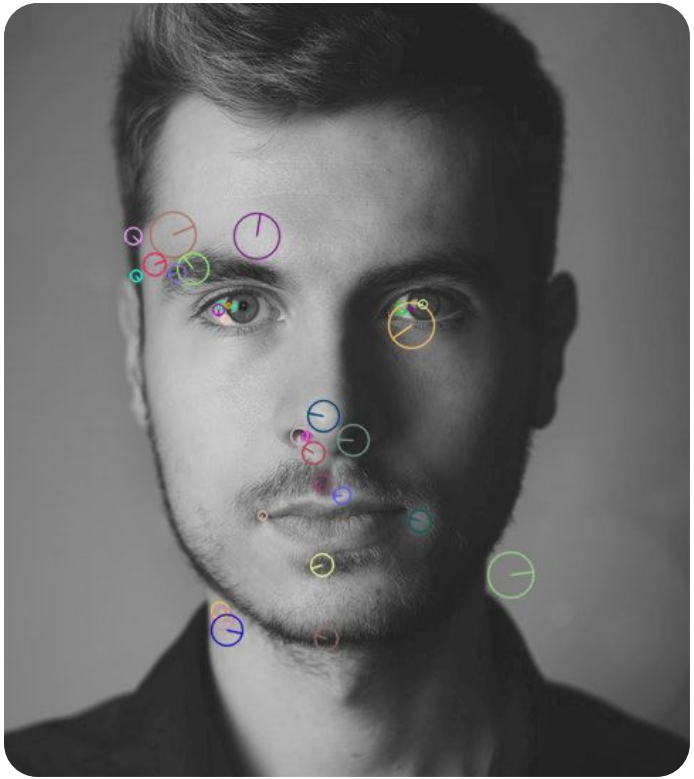
Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή



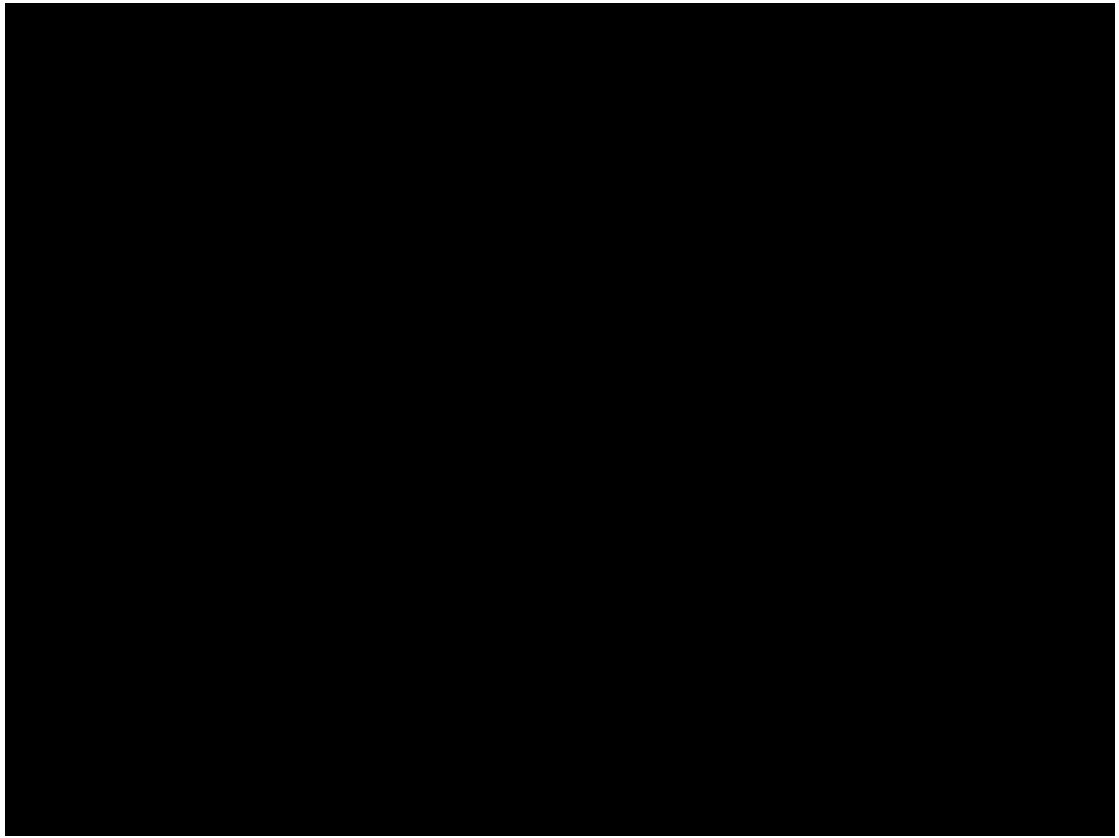
Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

«Δομή από Κίνηση (ΔαΚ) είναι η διαδικασία εκτίμησης μιας τρισδιάστατης δομής από μια ακολουθία δισδιάστατων εικόνων»

Δομή από Κίνηση
σε Πραγματικό Χρόνο
μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή



Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

> ***"Κάθε σύστημα που λειτουργεί και μπορεί να ανταποκριθεί σε γεγονότα εντός προβλέψιμων και συγκεκριμένων χρονικών περιορισμών"***

Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

Ας εξετάσουμε τα 2 ακόλουθα συστήματα ελέγχου ως παράδειγμα:

- Ο έλεγχος άρδευσης καλλιεργειών δεν χρειάζεται να πληροί κριτήρια πραγματικού χρόνου
- Ο έλεγχος πτήσης αεροπλάνου πρέπει να πληροί κριτήρια πραγματικού χρόνου

Τι θα γίνει αν υπάρξουν **2 δευτερόλεπτα καθυστέρησης:**

- Όταν ποτίζουμε καλλιέργειες;
- Όταν ελέγχουμε ένα αεροπλάνο;

Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

> ***«Μια διάταξη FPGA είναι ένα ολοκληρωμένο κύκλωμα σχεδιασμένο να διαμορφώνεται από έναν πελάτη ή έναν σχεδιαστή & μετά την κατασκευή του, ακόμη και κατά τη λειτουργία του στο πεδίο»***

Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

Ε και;!
Μπορώ και με το Arduino/Raspberry Pi μου!
Απλώς θα αλλάξω τον κώδικα...

Το **υλικό** που εκτελεί τον κώδικα **παραμένει** το **ίδιο** ανεξάρτητα από το πρόβλημα όταν χρησιμοποιούμε πλακέτες γενικής χρήσης.

Το υλικό ενός FPGA **αλλάζει** για την αντιμετώπιση κάθε προβλήματος.

Είναι ουσιαστικά ένα δυναμικό ASIC* με αποτέλεσμα γρήγορες αποδόσεις χωρίς διακυμάνσεις

*Application Specific Integrated Circuit

Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

Ε και;!
Μπορώ και με το Arduino/Raspberry Pi μου!
Απλώς θα αλλάξω τον κώδικα...

Έστω ότι πρέπει να προσθέσουμε δύο αριθμούς 5-bit (πχ 3+4)

Ένα Rasb Pi θα είχε κυμαινόμενη (αργή) απόδοση:

- Θα χρησιμοποιήσει τις πλησιέστερες μεταβλητές στα 5 bit (περιττοί υπολογισμοί)
- Θα έχει να διαχειριστεί διεργασίες ή διακοπές συστήματος για πιθανές εισόδους USB(ποντίκι, πληκτρολόγιο κ.α.)

Το FPGA θα έχει πάντα την ίδια (γρήγορη) απόδοση:

- Θα χρησιμοποιεί έναν αθροιστή 10 εισόδων (μέσω ενός συνόλου απο πολυπλέκτες)
- Θα εξάγει αποτελέσματα τόσο γρήγορα όσο η ηλεκτρική ενέργεια περνά μέσα από τους πολυπλέκτες!

Δομή από Κίνηση σε Πραγματικό Χρόνο μέσω διατάξεων FPGA

Μέρος 1: Εντοπισμός & Περιγραφή

Συνήθως, τα συστήματα Δομής από Κίνηση χωρίζουν αυτή τη διαδικασία σε τρία στάδια:

1. Εντοπισμός και εξαγωγή (περιγραφή) χαρακτηριστικών
2. Αντιστοίχιση χαρακτηριστικών και γεωμετρική επαλήθευση
3. Ανακατασκευή δομής και κίνησης

Σήμερα θα εξετάσουμε έργο το οποίο σχετίζεται με το πρώτο στάδιο.

ΤΙ ΑΚΡΙΒΩΣ ΕΝΤΟΠΙΖΟΥΜΕ ΚΑΙ ΠΕΡΙΓΡΑΦΟΥΜΕ;

χαρακτηριστικά σημεία

Η ΔαΚ(και άλλοι αλγόριθμοι υπολογιστικής όρασης/επεξεργασίας εικόνας) βασίζονται σε **χαρακτηριστικά σημεία εικόνας**.

Τι είναι αυτά τα χαρακτηριστικά σημεία;

Ο ορισμός ενός χαρακτηριστικού σημείου ποικίλλει. Γενικά είναι μια πληροφορία σχετικά με το περιεχόμενο μιας εικόνας (συνήθως σχετικά με το εάν μια συγκεκριμένη περιοχή της εικόνας πληροί ορισμένα κριτήρια)

Ανάλογα με τον τελικό στόχο, ένας ορισμός μπορεί να είναι καλύτερος από άλλους!

ΤΙ ΑΚΡΙΒΩΣ ΕΝΤΟΠΙΖΟΥΜΕ ΚΑΙ ΠΕΡΙΓΡΑΦΟΥΜΕ;

χαρακτηριστικά σημεία

Η ΔαΚ(και άλλοι αλγόριθμοι υπολογιστικής όρασης/επεξεργασίας εικόνας) βασίζονται σε **χαρακτηριστικά σημεία εικόνας**.

Τι είναι αυτά τα χαρακτηριστικά σημεία;

Ένα παράδειγμα ορισμού είναι:
"οποιοδήποτε pixel είναι κόκκινο"

Ένα άλλο παράδειγμα είναι:
«Ένα κόκκινο pixel που γειτονεύει ταυτόχρονα με ένα λευκό και ένα μπλε pixel"

Ποιο είναι το καλύτερο για να ανιχνεύσουμε τη σημαία των Φιλιππίνων σε ένα χωράφι με παπαρούνες;


Ito ay isang
pagsubok
www.bees.gr © Kostas Koutsoukos

ΤΙ ΑΚΡΙΒΩΣ ΕΝΤΟΠΙΖΟΥΜΕ ΚΑΙ ΠΕΡΙΓΡΑΦΟΥΜΕ;

Κατάλαβα τι είναι τα χαρακτηριστικά σημεία.
Τι είναι όμως η περιγραφή/εξαγωγή;

Ανιχνεύοντας μόνο γνωρίζουμε εάν ορισμένα pixel (ή μια περιοχή pixel) είναι χαρακτηριστικά σημεία (ή όχι). Αυτό είναι άχρηστο σε προβλήματα συνδυασμού πολλαπλών εικόνων (π.χ. δημιουργία πανοραμικής εικόνας).

Πώς γνωρίζουμε αν το σημείο που μόλις εντοπίσαμε είναι ένα από τα σημεία που εντοπίσαμε σε μια άλλη εικόνα; Η απάντηση είναι απλή. Πρέπει να δημιουργήσουμε μια περιγραφή για κάθε χαρακτηριστικό. Εξαγωγή σημαίνει περιγραφή των χαρακτηριστικών σημείων μας.

Όπως και με τα χαρακτηριστικά σημεία, ο ορισμός ενός περιγραφέα ποικίλλει και ο καλύτερος εξαρτάται από το πρόβλημα μας!

ΤΙ ΑΚΡΙΒΩΣ ΕΝΤΟΠΙΖΟΥΜΕ ΚΑΙ ΠΕΡΙΓΡΑΦΟΥΜΕ;

περιγραφείς

Κατάλαβα τι είναι τα χαρακτηριστικά σημεία.
Τι είναι όμως η περιγραφή/εξαγωγή;

Ένα παράδειγμα ορισμού ενος περιγραφέα είναι:

- Πόσοι από τους 8 γείτονες του εικονοστοιχείου έχουν μεγαλύτερη τιμή από το εικονοστοιχείο (σε κλίμακα του γκρι);
- Ο περιγραφέας θα είναι ένας αριθμός με τιμές από 0 έως 8

Ένα άλλο παράδειγμα ορισμού είναι:

- Συγκρίσεις αν το pixel είναι μεγαλύτερο από τα pixel ενός παραθύρου 5χ5 γύρω από το ίδιο.
- Ο περιγραφέας θα είναι ένα διάνυσμα μήκους 24 (τιμές είτε 0 είτε 1)

ΠΩΣ ΕΝΤΟΠΙΖΟΥΜΕ ΚΑΙ ΠΕΡΙΓΡΑΦΟΥΜΕ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΣΗΜΕΙΑ;

Τώρα που γνωρίζουμε τι είναι η ανίχνευση και η εξαγωγή, ποιες ακριβώς είναι οι επιλογές μας για την αντιμετώπιση του τμήματος ανίχνευσης και εξαγωγής της Δομής από Κίνηση;

Προφανώς δεν πρόκειται να εφεύρουμε ξανά τον τροχό, πολλοί διαφορετικοί αλγόριθμοι/μέθοδοι υπάρχουν ήδη εκεί έξω.

Για την εφαρμογή μας, αποφασίσαμε να επιλέξουμε τον αλγοριθμο ORB

ΤΙ ΚΑΘΙΣΤΑ ΕΝΑΝ ΑΛΓΟΡΙΘΜΟ ΚΑΛΟ;

επιλογές

Οι ανιχνευτές και οι περιγραφείς χαρακτηριστικών σημείων αξιολογούνται με βάση:

1. Ανθεκτικότητα σε αλλαγές κλίμακας (scale)
2. Ανθεκτικότητα σε διάτμηση (αλλαγή οπτικής γωνίας) (shear/viewport change)
3. Ανθεκτικότητα σε περιστροφές (rotation)
4. Ανθεκτικότητα σε θόρυβο (noise)
5. Ανθεκτικότητα σε θόλωμα (blur)
6. Ανθεκτικότητα σε αλλαγές φωτισμού (illumination change)
7. Επιδόσεις (χρόνος εκτέλεσης, παραλληλοποίηση, πόροι)

Διαισθητικά, όσο πιο ποιοτικα τα αποτελεσματα, τόσο μεγαλύτερος ο χρόνος εκτέλεσης.

Αλλά η ταχύτητα εκτέλεσης είναι υψίστης σημασίας για εμάς, καθώς θα χρειαστεί να εξάγουμε χαρακτηριστικά και περιγραφείς για ταχύτητες έως και 30 καρέ ανά δευτερόλεπτο για εισόδους 1920x1080 pixel!

Άρα θα θυσιάσουμε την ποιότητα των χαρακτηριστικών για χάρη της ταχύτητας, σωστά;

… όχι ακριβώς :)

ΓΙΑΤΙ ORB ΚΑΙ ΟΧΙ ΚΑΠΟΙΟΝ ΑΛΛΟ ΑΛΓΟΡΙΘΜΟ;

γιατι ORB;

Ο αλγόριθμος ORB είναι ένας από τους ταχύτερους αλγόριθμους. Συγκριτικά με τον αλγοριθμο SIFT (ο οποίος χρησιμοποιείται ως μέτρο συγκρισης στην βιβλιογραφία), ο ORB αποδίδει ανάλογα την περίπτωση:

- κατα 30% χειρότερα ή και καλύτερα του SIFT

Ο αλγόριθμος ORB εχει για της περιπτωση της ΔαΚ 3 πολυ σημαντικα οφέλη:

1. μικρό χρόνο εκτέλεσης **2.** παραληλοποιείται **3.** δεν είναι απαιτητικός σε πορους

Επίσης, δεν χρειάζεται να ικανοποιήσουμε όλα τα κριτήρια ανθεκτικότητας, καθώς η ΔαΚ λειτουργεί μέσω διαδοχικών καρέ (βίντεο). Καθώς θα εργαζόμαστε με 60 FPS, τα κριτήρια για τη συγκεκριμένη περίπτωση προβλήματος αλλάζουν ως εξής:

1. ~~Ανθεκτικότητα σε αλλαγές κλίμακας~~
2. ~~Ανθεκτικότητα σε διάτμηση (αλλαγή οπτικής γωνίας)~~
3. ~~Ανθεκτικότητα σε περιστροφές~~
4. Ανθεκτικότητα σε θόρυβο
5. Ανθεκτικότητα σε θόλωμα
6. ~~Ανθεκτικότητα σε αλλαγές φωτισμού~~

Έτσι, αφαιρώντας τα κομματια που σχετίζονται με τα προβληματα της περιστροφής και της κλίμακας, μπορούμε να κάνουμε τον αλγόριθμο ακόμα πιο γρήγορο!

ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

ORB συνοπτικά

Εντοπισμός

1. ~~Δημιουργία γκαουσιανής πυραμίδας~~
    a. Εντοπισμός FAST
    b. Φιλτράρισμα αποτελεσμάτων με Harris Response
    c. ~~Υπολογισμός γωνίας κεντροειδούς έντασης~~
    d. Φιλτράρισμα πυκνών χαρακτηριστικών σημείων
    e. Αγνόηση αποτελεσματων περιγραμματος

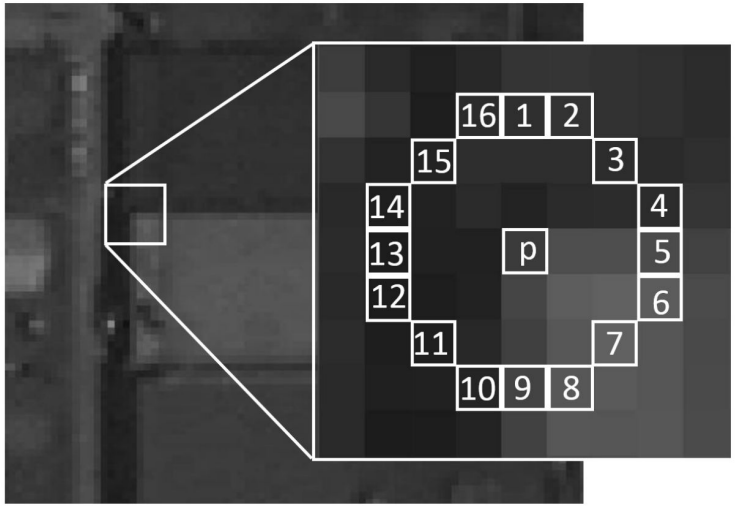
Περιγραφή

1. Διάνυσμα 256 boolean τιμών (κάθε τιμή = σύγκριση 2 pixel σε κλίμακα γκρι)
    a. ~~Με 12 προκαθορισμένες περιστροφές απο τις οποίες διαλέγουμε μια ανάλογα με την γωνία της κεντροειδούς έντασης~~

ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

FAST

"Αν N συνεχόμενα pixel είναι φωτεινότερα (ή πιο σκούρα) από το p + όριο (ή - όριο) τότε το p θεωρείται γωνία»



ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

Harris Resp.

> *"Ένα pixel θεωρείται γωνία εάν η τιμή Harris είναι μεγαλύτερη από 10000"*

$$Response = determinant(M) - k * trace(M)^2$$

- k = σταθερά σε διάστημα [0.04 , 0.06]
- M = μήτρα αυτοσυσχέτισης

$$M = \begin{bmatrix} gk(xx) & gk(xy) \\ gk(xy) & gk(yy) \end{bmatrix}$$

gk() = gaussian φίλτρο, x, y = μερικές παράγωγοι

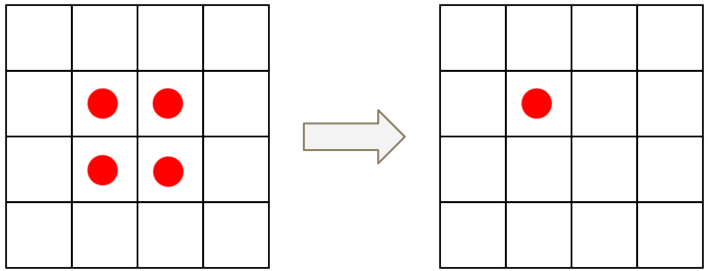
ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

corner peaks

Φιλτράρισμα αποτελεσμάτων έτσι ώστε τα χαρακτηριστικά σημεία να έχουν μια ελάχιστη απόσταση μεταξύ τους. Ας υποθέσουμε ότι έχουμε 4 pixel ως χαρακτηριστικά σημεία.

- Ένα φίλτρο dist=0 , θα έχει ως αποτέλεσμα την αναφορά και των 4 ως χαρακτηριστικών σημείων.

- Ένα φίλτρο dist=1 θα έχει ως αποτέλεσμα την αναφορά μόνο του 1ου ως χαρακτηριστικό σημείο.



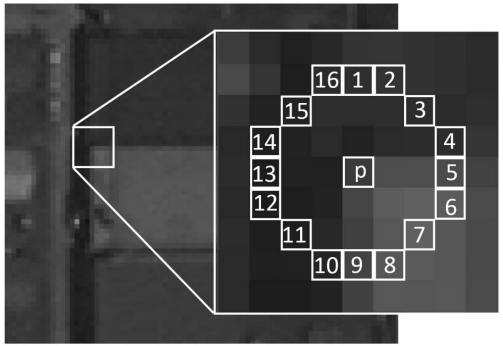
ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

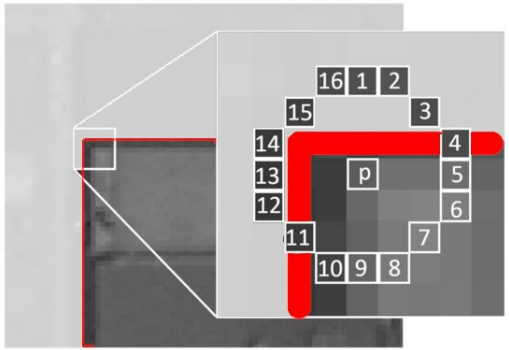
απόρριψη περιγράμματος

Χρειαζόμαστε πληροφορίες περιβάλλοντος (γειτονικά pixel) για να επισημάνουμε ένα pixel ως χαρακτηριστικό σημείο.

Όταν βρισκόμαστε κοντά στα όρια της εικόνας σημαίνει ότι αυτές οι πληροφορίες είναι εκτός πλαισίου (δεν υπάρχουν).

Ως αποτέλεσμα, όλα τα εικονοστοιχεία που βρίσκονται σε απόσταση μικρότερη του X από το περίγραμμα της εικόνας αγνοούνται ως αποτελέσματα.





ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ Ο ORB;

256 ζευγάρια pixel

Για κάθε χαρακτηριστικό σημείο, ο περιγραφέας μας είναι ένα διάνυσμα 256 boolean τιμών (συγκρίσεων εικονοστοιχείων)

Ας χρησιμοποιήσουμε ένα μικρότερο παράδειγμα 2 συγκρίσεων. Αυτό σημαίνει ότι ο περιγραφέας μας θα είναι ένα διάνυσμα μήκους 2 -> [x1 x2]

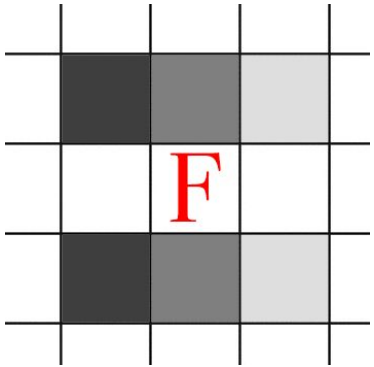
Θα θεωρήσουμε ως

- 1 (αληθές) όταν ο πρώτος αριθμός είναι >= από τον 2ο, και
- 0 (ψευδές) διαφορετικά.

Οι συντεταγμένες των συγκρίσεων μας είναι:

1. [-1, -1] με [1, 1]
2. [-1, 1] με [1, -1]

Αυτό σημαίνει ότι για το παράδειγμα στα δεξιά ο περιγραφέας μας είναι: [0, 1]



ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

Υπάρχει ένα σύνολο πραγμάτων που πρέπει να κατανοήσουμε πριν προχωρήσουμε στη μεταφορά του ORB σε διατάξεις FPGA, αυτά είναι:

- Λογική υλικού (μη-ακολουθιακή εκτέλεση, ρολόγια)
- Κυλιόμενα παράθυρα (έλλειψη ολοκληρωτικής εικόνας δεδομένων)
- Buffers

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

#υλικό

Το πρώτο πράγμα που πρέπει να καταλάβουμε σχετικά με τις υλοποιήσεις υλικού είναι ότι δεν ακολουθούν τη ακολουθιακή λογική του λογισμικού!

1. Όλα συμβαίνουν ταυτόχρονα

Παράδειγμα

$$A_{next} = B_{previous}$$
$$B_{next} = C_{previous}$$
$$C_{next} = A_{previous}$$

Στο λογισμικό το C_{next} θα ήταν $B_{previous}$ αλλά στο υλικό θα είναι $A_{previous}$

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

#υλικό

Το δεύτερο πράγμα που πρέπει να καταλάβουμε σχετικά με τις υλοποιήσεις υλικού είναι ότι η υλοποίηση «εκτελείται» συνεχώς

2. Όλα συμβαίνουν συνεχώς, σε αντίθεση με το λογισμικό

Παραδειγμα

$$A = B + C$$

Στο λογισμικό η προσθήκη θα εκτελεστεί και το πρόγραμμα θα τερματιστεί. Στην περιγραφή υλικού, αυτός θα ήταν ένας αθροιστής που θα εξάγει συνεχώς το αποτέλεσμα. Τη στιγμή που κάποια από τις 2 τιμές αλλάξει, το A θα αλλάξει τόσο γρήγορα όσο το ρεύμα μπορεί να μεταδοθεί μέσω του αθροιστή.

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

Το τρίτο πράγμα που πρέπει να καταλάβουμε σχετικά με τις υλοποιήσεις υλικού είναι ότι οι είσοδοι και αντίστοιχα οι έξοδοι μας διέπονται κυρίως από συμβάντα ρολογιού

3. Τα δεδομένα συνήθως αλλάζουν στις αλλαγές του ρολογιού

Παράδειγμα

$$A = B + C$$

Εάν το A χρειάζεται ένα X χρόνο για να απεικονίσει σωστά το αποτέλεσμα του B + C μετά από αυτές τις 2 αλλαγές, σε ποια χρονική στιγμή πρέπει να καταγράψω το A για να ξέρω ότι έχω τη σωστή τιμή της άθροισης;

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

Η εργασία με δεδομένα “on the fly” (δεν υπάρχουν διαθέσιμα όλα τα δεδομένα) είναι ένα άλλο χαρακτηριστικό των υλοποιήσεων υλικού.

Για παράδειγμα

όταν εφαρμόζουμε τροποποιήσεις σε μια εικόνα, δεν μπορουμε να έχουμε όλα τα δεδομένα διαθέσιμα, επειδή τις περισσότερες φορές δεν έχουμε αρκετό υλικό για να τα αποθηκεύσουμε.

Οι υλοποιήσεις λογισμικού χρησιμοποιούν εξωτερική μνήμη με αποτέλεσμα τη μεταφορά λειτουργιών από και προς αυτήν τη μνήμη (συνήθως RAM)

Δεν μπορούμε να το κάνουμε αυτό σε υλοποιήσεις υλικού για τους παρακάτω λόγους:

1. κριτήρια απόδοσης που συνήθως πρέπει να πληρούμε, ή
2. είναι αδύνατο λόγω περιορισμών υλικού

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

κυλιόμενα παράθυρα

Η εργασία με δεδομένα “on the fly” (δεν υπάρχουν διαθέσιμα όλα τα δεδομένα) είναι ένα άλλο χαρακτηριστικό των υλοποιήσεων υλικού.

Ως αποτέλεσμα, όταν πρόκειται για εργασίες που σχετίζονται με την εικόνα, εργαζόμαστε με "κυλιόμενα παράθυρα". Αντί να δουλεύουμε με όλη την εικόνα, εργαζόμαστε με τα δεδομένα "όπως αυτά έρχονται".

Ας υποθέσουμε ότι θέλουμε να κάνουμε το παρακάτω:

- Κάθε pixel να είναι ένα άθροισμα των γειτόνων του κατα την έξοδο

Σε μια υλοποίηση λογισμικού θα είχαμε έναν πίνακα 2d (matrix) και θα υπολογίζαμε διαδοχικά κάθε pixel το αποτέλεσμα.

Τι κάνουμε για να λύσουμε αυτό το πρόβλημα από πλευράς υλικού όταν έρχονται συνεχώς πληροφορίες;

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

κυλιόμενα παράθυρα

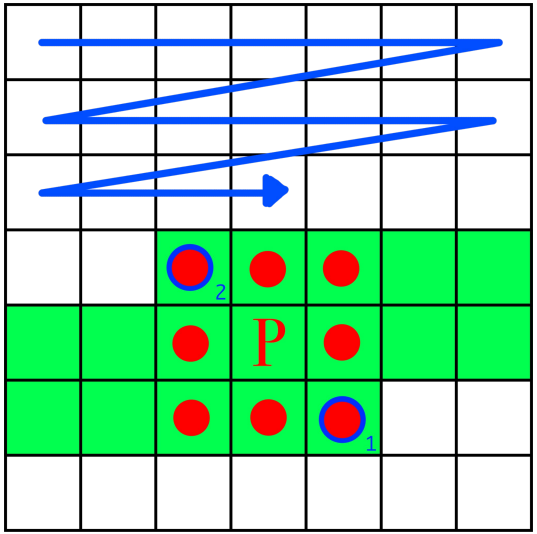
Έστω ότι χρειαζόμαστε πληροφορίες από μια περιοχή 3x3 (κόκκινες κουκίδες) γύρω από ένα pixel(P). Ενα παραδειγμα εικόνας συνολικού μεγέθους 7x7 μοιάζει με το παρακάτω.

Τα pixel έρχονται ως ροή (μπλε γραμμή). Που σημαίνει ότι σε κάθε είσοδο pixel (μπλε pixel 2), μπορούμε να υπολογίσουμε το pixel που λάβαμε πριν από 8 επαναλήψεις (P).

Σημαίνει επίσης ότι για να κάνουμε τη λειτουργία πρέπει να παρακολουθούμε pixel έως και 16 επαναλήψεις πριν (μέχρι το μπλε pixel 1).

Οι (πράσινες) πληροφορίες που παρακολουθούμε συνεχώς είναι το κυλιόμενο παράθυρό μας. Αυτο ειναι :

- πολύ πιο αποτελεσματικό από το να περιμένουμε ολόκληρη την εικόνα για να ξεκινήσουμε τους υπολογισμούς
- λιγότερο απαιτητικό σε πόρους υλικού



ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

Τι είναι ένας buffer;

Ένας buffer είναι ουσιαστικά μια υλοποίηση υλικού που ακολουθεί τη λογική της δομής δεδομένων λογισμικού.

Μπορούμε να το σκεφτούμε ως έναν καναπέ με καθορισμένο/πεπερασμένο αριθμό θέσεων. Κάθε φορά που κάποιος μπαίνει και κάθεται από τη δεξιά πλευρά, σπρώχνει τους πάντες να μετακινήσουν ένα κάθισμα προς τα αριστερά και με αποτέλεσμα αυτός που κάθεται στο πιο αριστερό κάθισμα, προφανώς πρέπει να βγει από τον καναπέ!

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

Η υλοποίηση μας αποτελείται από 8 αρχεία

- Container
- Linker
- Το κυρίως orb αρχείο
- Grayscale
- 4 τύπους buffers αρχείων

ΜΕΤΑΦΕΡΟΝΤΑΣ ΤΟΝ ORB ΣΕ ΔΙΑΤΑΞΕΙΣ FPGA

ιεραρχία

- Container
  - Linker
    - Buffer 2
    - Buffer 3
    - Buffer 4
    - Grayscale
    - Orb
      - Buffer 1

Container
Linker
Buffer 2
Buffer 3
Buffer 4
Grayscale
Orb
Buffer 1

αρχεία buffer

Έχουμε 4 παραλλαγές από buffers

Buffer 1
Δέχεται (generic) είσοδο & εξάγει δεδομένα όταν
- Το ρολόι pixel πραγματοποιεί ανοδικό παλμό
- Είναι ενεργή περίοδος pixel (no blanking period)

Buffer 2
Δέχεται (boolean) είσοδο & εξάγει δεδομένα όταν
- Το ρολόι pixel πραγματοποιεί ανοδικό παλμό

Buffer 3
Δέχεται (boolean) είσοδο & εξάγει δεδομένα όταν
- Το ρολόι pixel πραγματοποιεί καθοδικό παλμό
- Είναι ενεργή περίοδος pixel (no blanking period)

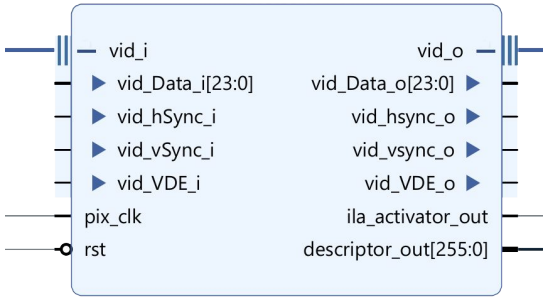
Buffer 4
Δέχεται (generic) είσοδο & εξάγει δεδομένα όταν
- Το ρολόι pixel πραγματοποιεί καθοδικό παλμό

Container
Linker
Buffer 2
Buffer 3
Buffer 4
Grayscale
Orb
Buffer 1

orb.vhd

Ποια ειναι η λειτουργια του;

1. Λαμβάνει τις εισόδους και τις μεταφέρει στο αρχείο linker
2. Ταυτόχρονα λαμβάνει δεδομένα από το αρχείο linker και τα εξάγει



Container
Linker
Buffer 2
Buffer 3
Buffer 4
Grayscale
Orb
Buffer 1

linker.vhd

Ποια ειναι η λειτουργια του;

1. Λαμβάνει δεδομένα από το αρχείο container και τα στέλνει στο αρχείο grayscale
2. Λαμβάνει την έξοδο του αρχείου grayscale και την μεταφέρει στο αρχείο core
3. Λαμβάνει τα αποτελέσματα του αρχείου core & τα φιλτράρει
4. Εξάγει τα φιλτραρισμένα δεδομένα πίσω στο αρχείο container

Τι είδους φιλτράρισμα κάνει;

- Αγνοεί χαρ. σημεία πολύ δίπλα σε άλλα χαρ. σημεία
- Ελέγχει για pixel περιγράμματος (αγνοεί χαρ. σημεία στο περίγραμμα της εικόνας)
- Εξάγει τα σωστά δεδομένα όταν το αρχείο core περνά από περιόδους κενού ή όταν τα υπόλοιπα 2 φίλτρα είναι ενεργά για να παρέχει σωστή έξοδο HDMI

Οι έξοδοι του αρχείου linker είναι

- Ένα pixel ανά κύκλο ρολογιού σε κλίμακα του γκρι (κόκκινο εάν το pixel είναι χαρακτηριστικό σημείο)
- Ένας περιγραφέας (0 * 256 όταν δεν έχουμε χαρ σημείο)
- Ένα σήμα boolean που χαρακτηριζει αν το εξαγόμενο pixel είναι χαρακτηριστικό σημείο

Container
Linker
Buffer 2
Buffer 3
Buffer 4
Grayscale
Orb
Buffer 1

rgb2gray.vhd

Ποια ειναι η λειτουργια του;

1. Λαμβάνει είσοδο 24 bit (8 bit ανά χρώμα).
2. Μετατρέπεται σε κλίμακα του γκρι 8 bit
3. Εξάγει 8 bit κλίμακα του γκρι

Container
Linker
Buffer 2
Buffer 3
Buffer 4
Grayscale
Orb
Buffer 1

core.vhd

Ποια είναι η λειτουργία του;

1. Λαμβάνει είσοδο σε κλίμακα του γκρι
2. Υπολογίζει εάν ένα pixel απο 24978 κυκλους πριν (28618 κύκλους πριν συμπεριλαμβανομένων των περιόδων κενού) ήταν χαρακτηριστικό σημειο
3. Υπολογίζει τους περιγραφείς (χαρακτηριστικό σημείων και μη)
4. Εξάγει
   a. την τιμή της κλίμακας του γκρι (κόκκινο εάν χαρ σημείο)
   b. Τον περιγραφέα (bool διάνυσμα μήκους 256)
   c. Ένα σήμα boolean που χαρακτηριζει αν το εξαγόμενο pixel είναι χαρακτηριστικό σημείο

Πώς υπολογίζονται τα χαρακτ/κα σημεία και οι περιγραφείς τους;

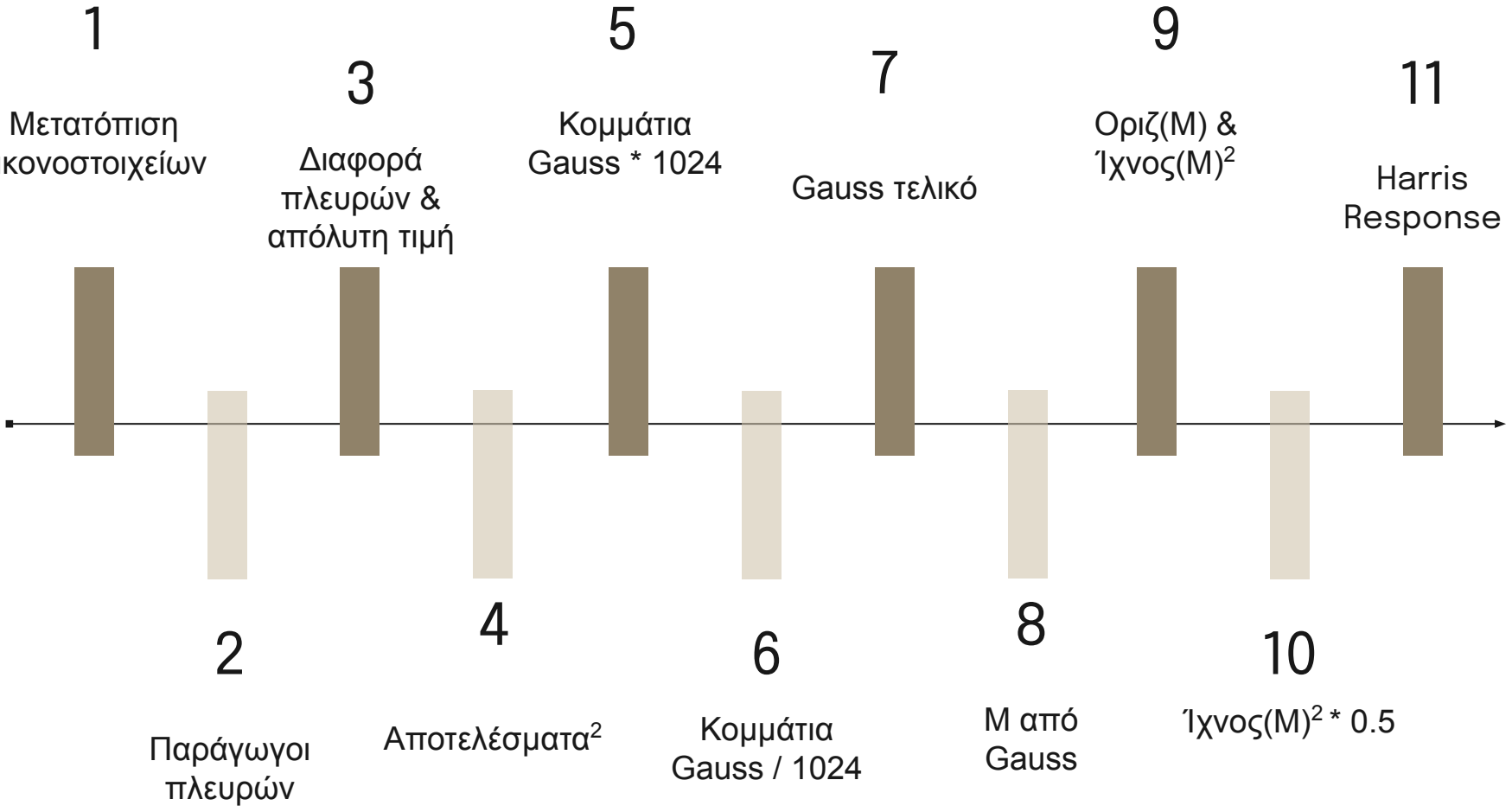
Η διαδικασία λήψης απόφασης για ένα χαρ. σημείο και εξαγωγής του προκύπτει από ένα σύνολο υπολογισμών που απαιτούν συνολικά 11 παλμούς ρολογιού για να ολοκληρωθούν. Αυτό σημαίνει 5 κύκλους ρολογιού από τη στιγμή που έχουμε όλα τα απαιτούμενα pixel

παράθυρο

k27_27	k27_26	k27_25	k27_24	k27_23	k27_22	k27_21	k27_20	k27_19	k27_18	k27_17	k27_16	k27_15	k27_14	k27_13	k27_12	k27_11	k27_10	k27_9	k27_8	k27_7	k27_6	k27_5	k27_4	k27_3	k27_2	k27_1
k26_27	k26_26	k26_25	k26_24	k26_23	k26_22	k26_21	k26_20	k26_19	k26_18	k26_17	k26_16	k26_15	k26_14	k26_13	k26_12	k26_11	k26_10	k26_9	k26_8	k26_7	k26_6	k26_5	k26_4	k26_3	k26_2	k26_1
k25_27	k25_26	k25_25	k25_24	k25_23	k25_22	k25_21	k25_20	k25_19	k25_18	k25_17	k25_16	k25_15	k25_14	k25_13	k25_12	k25_11	k25_10	k25_9	k25_8	k25_7	k25_6	k25_5	k25_4	k25_3	k25_2	k25_1
k24_27	k24_26	k24_25	k24_24	k24_23	k24_22	k24_21	k24_20	k24_19	k24_18	k24_17	k24_16	k24_15	k24_14	k24_13	k24_12	k24_11	k24_10	k24_9	k24_8	k24_7	k24_6	k24_5	k24_4	k24_3	k24_2	k24_1
k23_27	k23_26	k23_25	k23_24	k23_23	k23_22	k23_21	k23_20	k23_19	k23_18	k23_17	k23_16	k23_15	k23_14	k23_13	k23_12	k23_11	k23_10	k23_9	k23_8	k23_7	k23_6	k23_5	k23_4	k23_3	k23_2	k23_1
k22_27	k22_26	k22_25	k22_24	k22_23	k22_22	k22_21	k22_20	k22_19	k22_18	k22_17	k22_16	k22_15	k22_14	k22_13	k22_12	k22_11	k22_10	k22_9	k22_8	k22_7	k22_6	k22_5	k22_4	k22_3	k22_2	k22_1
k21_27	k21_26	k21_25	k21_24	k21_23	k21_22	k21_21	k21_20	k21_19	k21_18	k21_17	k21_16	k21_15	k21_14	k21_13	k21_12	k21_11	k21_10	k21_9	k21_8	k21_7	k21_6	k21_5	k21_4	k21_3	k21_2	k21_1
k20_27	k20_26	k20_25	k20_24	k20_23	k20_22	k20_21	k20_20	k20_19	k20_18	k20_17	k20_16	k20_15	k20_14	k20_13	k20_12	k20_11	k20_10	k20_9	k20_8	k20_7	k20_6	k20_5	k20_4	k20_3	k20_2	k20_1
k19_27	k19_26	k19_25	k19_24	k19_23	k19_22	k19_21	k19_20	k19_19	k19_18	k19_17	k19_16	k19_15	k19_14	k19_13	k19_12	k19_11	k19_10	k19_9	k19_8	k19_7	k19_6	k19_5	k19_4	k19_3	k19_2	k19_1
k18_27	k18_26	k18_25	k18_24	k18_23	k18_22	k18_21	k18_20	k18_19	k18_18	k18_17	k18_16	k18_15	k18_14	k18_13	k18_12	k18_11	k18_10	k18_9	k18_8	k18_7	k18_6	k18_5	k18_4	k18_3	k18_2	k18_1
k17_27	k17_26	k17_25	k17_24	k17_23	k17_22	k17_21	k17_20	k17_19	k17_18	k17_17	k17_16	k17_15	k17_14	k17_13	k17_12	k17_11	k17_10	k17_9	k17_8	k17_7	k17_6	k17_5	k17_4	k17_3	k17_2	k17_1
k16_27	k16_26	k16_25	k16_24	k16_23	k16_22	k16_21	k16_20	k16_19	k16_18	k16_17	k16_16	k16_15	k16_14	k16_13	k16_12	k16_11	k16_10	k16_9	k16_8	k16_7	k16_6	k16_5	k16_4	k16_3	k16_2	k16_1
k15_27	k15_26	k15_25	k15_24	k15_23	k15_22	k15_21	k15_20	k15_19	k15_18	k15_17	k15_16	k15_15	k15_14	k15_13	k15_12	k15_11	k15_10	k15_9	k15_8	k15_7	k15_6	k15_5	k15_4	k15_3	k15_2	k15_1
k14_27	k14_26	k14_25	k14_24	k14_23	k14_22	k14_21	k14_20	k14_19	k14_18	k14_17	k14_16	k14_15	k14_14	k14_13	k14_12	k14_11	k14_10	k14_9	k14_8	k14_7	k14_6	k14_5	k14_4	k14_3	k14_2	k14_1
k13_27	k13_26	k13_25	k13_24	k13_23	k13_22	k13_21	k13_20	k13_19	k13_18	k13_17	k13_16	k13_15	k13_14	k13_13	k13_12	k13_11	k13_10	k13_9	k13_8	k13_7	k13_6	k13_5	k13_4	k13_3	k13_2	k13_1
k12_27	k12_26	k12_25	k12_24	k12_23	k12_22	k12_21	k12_20	k12_19	k12_18	k12_17	k12_16	k12_15	k12_14	k12_13	k12_12	k12_11	k12_10	k12_9	k12_8	k12_7	k12_6	k12_5	k12_4	k12_3	k12_2	k12_1
k11_27	k11_26	k11_25	k11_24	k11_23	k11_22	k11_21	k11_20	k11_19	k11_18	k11_17	k11_16	k11_15	k11_14	k11_13	k11_12	k11_11	k11_10	k11_9	k11_8	k11_7	k11_6	k11_5	k11_4	k11_3	k11_2	k11_1
k10_27	k10_26	k10_25	k10_24	k10_23	k10_22	k10_21	k10_20	k10_19	k10_18	k10_17	k10_16	k10_15	k10_14	k10_13	k10_12	k10_11	k10_10	k10_9	k10_8	k10_7	k10_6	k10_5	k10_4	k10_3	k10_2	k10_1
k9_27	k9_26	k9_25	k9_24	k9_23	k9_22	k9_21	k9_20	k9_19	k9_18	k9_17	k9_16	k9_15	k9_14	k9_13	k9_12	k9_11	k9_10	k9_9	k9_8	k9_7	k9_6	k9_5	k9_4	k9_3	k9_2	k9_1
k8_27	k8_26	k8_25	k8_24	k8_23	k8_22	k8_21	k8_20	k8_19	k8_18	k8_17	k8_16	k8_15	k8_14	k8_13	k8_12	k8_11	k8_10	k8_9	k8_8	k8_7	k8_6	k8_5	k8_4	k8_3	k8_2	k8_1
k7_27	k7_26	k7_25	k7_24	k7_23	k7_22	k7_21	k7_20	k7_19	k7_18	k7_17	k7_16	k7_15	k7_14	k7_13	k7_12	k7_11	k7_10	k7_9	k7_8	k7_7	k7_6	k7_5	k7_4	k7_3	k7_2	k7_1
k6_27	k6_26	k6_25	k6_24	k6_23	k6_22	k6_21	k6_20	k6_19	k6_18	k6_17	k6_16	k6_15	k6_14	k6_13	k6_12	k6_11	k6_10	k6_9	k6_8	k6_7	k6_6	k6_5	k6_4	k6_3	k6_2	k6_1
k5_27	k5_26	k5_25	k5_24	k5_23	k5_22	k5_21	k5_20	k5_19	k5_18	k5_17	k5_16	k5_15	k5_14	k5_13	k5_12	k5_11	k5_10	k5_9	k5_8	k5_7	k5_6	k5_5	k5_4	k5_3	k5_2	k5_1
k4_27	k4_26	k4_25	k4_24	k4_23	k4_22	k4_21	k4_20	k4_19	k4_18	k4_17	k4_16	k4_15	k4_14	k4_13	k4_12	k4_11	k4_10	k4_9	k4_8	k4_7	k4_6	k4_5	k4_4	k4_3	k4_2	k4_1
k3_27	k3_26	k3_25	k3_24	k3_23	k3_22	k3_21	k3_20	k3_19	k3_18	k3_17	k3_16	k3_15	k3_14	k3_13	k3_12	k3_11	k3_10	k3_9	k3_8	k3_7	k3_6	k3_5	k3_4	k3_3	k3_2	k3_1
k2_27	k2_26	k2_25	k2_24	k2_23	k2_22	k2_21	k2_20	k2_19	k2_18	k2_17	k2_16	k2_15	k2_14	k2_13	k2_12	k2_11	k2_10	k2_9	k2_8	k2_7	k2_6	k2_5	k2_4	k2_3	k2_2	k2_1
k1_27	k1_26	k1_25	k1_24	k1_23	k1_22	k1_21	k1_20	k1_19	k1_18	k1_17	k1_16	k1_15	k1_14	k1_13	k1_12	k1_11	k1_10	k1_9	k1_8	k1_7	k1_6	k1_5	k1_4	k1_3	k1_2	k1_1

Σε κάθε κύκλο του pixel clock -> (*εισάγεται νέο k1_1*) (*εξάγεται το k14_19*) (*εφαρμογή FAST στο k14_14*) (*παράγεται ο περιγραφέας του k14_14*)

χρονισμός παλμών



ΕΛΕΓΧΟΝΤΑΣ ΤΗΝ ΥΛΟΠΟΙΗΣΗ ΜΑΣ

επικύρωση

Επικυρώσαμε την υλοποίησή μας έναντι της εφαρμογής ORB της βιβλιοθήκης Scikit-Image της γλώσσας python.

Δημιουργήσαμε προσαρμοσμένες εικόνες 1920x1080 με ακρίβεια pixel, για να συγκρίνουμε εάν τα χαρακτηριστικά και οι αντίστοιχοι περιγραφείς ταυτίζονται.

Το αρχικό σχέδιο ήταν να τροφοδοτηθούν αυτές οι εικόνες στην βιβλιοθήκη scikit-image και στη συνέχεια να προβληθούν σε λειτουργία πληρης οθόνης (στο εξωτερικό monitor)

Αποδεικνύεται ότι το λειτουργικό σύστημα κάνει πολλές προσαρμογές εάν η ανάλυση της οθόνης δεν είναι η ίδια της εξαγόμενης απο την πλακέτα (1920 x 1080) και ως αποτέλεσμα κάναμε 2 επαληθεύσεις , δηλαδή:

Σε εξωτερικη οθονη διαφορετικης ανάλυσης απο 1920x1080 κάναμε print-screen της κατάστασης η οποία εμφανιζόταν στο εξωτερικό monitor (αποτελέσματα της πλακέτας), και την τροφοδοτούσαμε στην υλοποίηση της βιβλιοθήκης Scikit-Image

Σε οθονη με αναλυση 1920x1080 ακολουθήσαμε το αρχικό σχέδιο

ΑΠΟΤΕΛΕΣΜΑΤΑ;

ΗΤΑΝ ΟΛΑ ΤΟΣΟ ΕΥΚΟΛΑ;

επιπλοκές

Όπως είδαμε, η είσοδος για το DEMO μας είναι μια πηγή HDMI που προέρχεται από το φορητό υπολογιστή.

Το έργο ξεκίνησε αρχικά με τη χρήση της κάμερας MIPI. Το πρόβλημα που παρουσιάστηκε είναι ότι χαρακτηριστικά σημεία εντοπίζονται παντού λόγω του θορύβου της κάμερας.

Για να αντιμετωπίσουμε το εν λόγω πρόβλημα, υπάρχουν 2 λύσεις:

- Ο γρήγορος και ‘μπακάλικος’ τρόπος είναι να ορίσουμε το όριο του FAST σε μεγαλύτερη τιμή (30 έως 40). Με αυτόν τον τρόπο ο θόρυβος εξαλείφεται, αλλά ταυτόχρονα ένας μεγάλος όγκος χαρακτηριστικών σημείων χάνεται.

- Ο σωστός τρόπος είναι να εφαρμόσουμε ένα φίλτρο θολωματος σε όλα τα pixel του παραθύρου (kernel) πριν τα χρησιμοποιήσουμε. Αλλά …

ΠΡΟΒΛΗΜΑΤΑ;

επιπλοκές

...η εφαρμογή ενός φίλτρου Gauss και στις 727 τιμές προυποθετει την υλοποίηση ενός νέου αρχείου ανάμεσα των rgb2gray και core.

1. Προσπαθήσαμε να το αποφύγουμε αυτό (για την εμπρόθεσμη παράδοσης της εργασίας) χρησιμοποιώντας ένα φίλτρο 5x5 για τα 12 pixel που απαιτούνται για την ανίχνευση FAST και συνειδητοποιήσαμε ότι παρόλο που αυτό μείωσε την ευαισθησία στον θόρυβο, δεν τον εξαφάνισε πλήρως.

θόρυβος

k19_20	k19_19	k19_18	k19_17	k19_16	k19_15	k19_14	k19_13	k19_12	k19_11	k19_10
k18_20	k18_19	k18_18	k18_17	k18_16	k18_15	k18_14	k18_13	k18_12	k18_11	k18_10
k17_20	k17_19	k17_18	k17_17	k17_16	k17_15	k17_14	k17_13	k17_12	k17_11	k17_10
k16_20	k16_19	k16_18	k16_17	k16_16	k16_15	k16_14	k16_13	k16_12	k16_11	k16_10
k15_20	k15_19	k15_18	k15_17	k15_16	k15_15	k15_14	k15_13	k15_12	k15_11	k15_10
k14_20	k14_19	k14_18	k14_17	k14_16	k14_15	k14_14	k14_13	k14_12	k14_11	k14_10
k13_20	k13_19	k13_18	k13_17	k13_16	k13_15	k13_14	k13_13	k13_12	k13_11	k13_10
k12_20	k12_19	k12_18	k12_17	k12_16	k12_15	k12_14	k12_13	k12_12	k12_11	k12_10
k11_20	k11_19	k11_18	k11_17	k11_16	k11_15	k11_14	k11_13	k11_12	k11_11	k11_10
k10_20	k10_19	k10_18	k10_17	k10_16	k10_15	k10_14	k10_13	k10_12	k10_11	k10_10

k19_20	k19_19	k19_18	k19_17	k19_16	k19_15	k19_14	k19_13	k19_12	k19_11	k19_10
k18_20	k18_19	k18_18	k18_17	k18_16	k18_15	k18_14	k18_13	k18_12	k18_11	k18_10
k17_20	k17_19	k17_18	k17_17	k17_16	k17_15	k17_14	k17_13	k17_12	k17_11	k17_10
k16_20	k16_19	k16_18	k16_17	k16_16	k16_15	k16_14	k16_13	k16_12	k16_11	k16_10
k15_20	k15_19	k15_18	k15_17	k15_16	k15_15	k15_14	k15_13	k15_12	k15_11	k15_10
k14_20	k14_19	k14_18	k14_17	k14_16	k14_15	k14_14	k14_13	k14_12	k14_11	k14_10
k13_20	k13_19	k13_18	k13_17	k13_16	k13_15	k13_14	k13_13	k13_12	k13_11	k13_10
k12_20	k12_19	k12_18	k12_17	k12_16	k12_15	k12_14	k12_13	k12_12	k12_11	k12_10
k11_20	k11_19	k11_18	k11_17	k11_16	k11_15	k11_14	k11_13	k11_12	k11_11	k11_10
k10_20	k10_19	k10_18	k10_17	k10_16	k10_15	k10_14	k10_13	k10_12	k10_11	k10_10

k19_20	k19_19	k19_18	k19_17	k19_16	k19_15	k19_14	k19_13	k19_12	k19_11	k19_10
k18_20	k18_19	k18_18	k18_17	k18_16	k18_15	k18_14	k18_13	k18_12	k18_11	k18_10
k17_20	k17_19	k17_18	k17_17	k17_16	k17_15	k17_14	k17_13	k17_12	k17_11	k17_10
k16_20	k16_19	k16_18	k16_17	k16_16	k16_15	k16_14	k16_13	k16_12	k16_11	k16_10
k15_20	k15_19	k15_18	k15_17	k15_16	k15_15	k15_14	k15_13	k15_12	k15_11	k15_10
k14_20	k14_19	k14_18	k14_17	k14_16	k14_15	k14_14	k14_13	k14_12	k14_11	k14_10
k13_20	k13_19	k13_18	k13_17	k13_16	k13_15	k13_14	k13_13	k13_12	k13_11	k13_10
k12_20	k12_19	k12_18	k12_17	k12_16	k12_15	k12_14	k12_13	k12_12	k12_11	k12_10
k11_20	k11_19	k11_18	k11_17	k11_16	k11_15	k11_14	k11_13	k11_12	k11_11	k11_10
k10_20	k10_19	k10_18	k10_17	k10_16	k10_15	k10_14	k10_13	k10_12	k10_11	k10_10

Για όλα τα pixel (13) της ανίχνευσης FAST, αυτό σημαίνει για:

- 12 pixel (του κύκλου σύγκρισης)
- 1 pixel (το κεντρικό pixel)

παίρνουμε μια τιμή Gaussian blur από ένα παράθυρο 5x5.

Αυτό απαιτεί + 3 μέτωπα παλμών ρολογιού pixel (1,5 κύκλος ρολογιού) για να υπολογιστεί εάν το σημείο ταξινομείται ως χαρακτηριστικό σημείο ή όχι.

Αυτό συμβαίνει επειδή πρέπει να αθροίσουμε 25 τιμές και να πάρουμε τον μέσο όρο. Αυτό γίνεται:

1. Ομαδοποιώντας τα 25 εικονοστοιχεία σε γκρουπ των 5 και λήψη ενός αθροίσματος για κάθε γρουπ -> Αυτό μας δίνει 5 τιμές/μεταβλητές
2. Αθροίζουμε τις τιμες των 5 γκρουπ του βηματος 1
3. Παίρνουμε τον μέσο όρο

ΠΡΟΒΛΗΜΑΤΑ;

επιπλοκές

...η εφαρμογή ενός φίλτρου Gauss και στις 727 τιμές προυποθετει την υλοποίηση ενός νέου αρχείου ανάμεσα των rgb2gray και core.

1. Προσπαθήσαμε να το αποφύγουμε αυτό (για την εμπρόθεσμη παράδοσης της εργασίας) χρησιμοποιώντας ένα φίλτρο 5x5 για τα 12 pixel που απαιτούνται για την ανίχνευση FAST και συνειδητοποιήσαμε ότι παρόλο που αυτό μείωσε την ευαισθησία στον θόρυβο, δεν τον εξαφάνισε πλήρως.

2. Επομένως μεταβαίνουμε στη χρήση HDMI ως πηγή εισόδου (αλλά η υλοποίηση ενός φίλτρου Gauss πριν το αρχείο core, είναι ο σωστός τρόπος για την πραγματική εφαρμογή σε παραγωγικά περιβάλλοντα)

Ένας άλλος τρόπος αντιμετώπισης, είναι να θεωρήσουμε την είσοδο ως εξωτερικό σύστημα και να επιβάλουμε/απαιτήσουμε να εισέρχεται στο FPGA άνευ θορύβου.

πόροι

ΥΛΙΚΟ

- Digilent Zybo Zynq 7020 FPGA
- Digilent Pcam 5C
- CAMPLUS παραμετροποιήσιμο τροφοδοτικό
- Λάπτοπ
- Εξωτερικό Monitor
- 2 HDMI καλώδια
- USB to micro usb καλώδιο
- Usb-c multi αντάπτορας

ΛΟΓΙΣΜΙΚΟ

- Windows 11
- Vivado
- Vitis
- Visual studio code + VHDL plugin
- Jetbrains PyCharm για SciImage, Testbench & Git
- GitLab
- Google Docs, Sheets & Slides
- Putty

Μελλοντική Δουλειά

Αυτό που διαδέχεται την τρέχουσα υλοποίηση ως φυσικό επακόλουθο είναι το 2ο μέρος της διαδικασίας “Δομή από Κίνηση”. Το οποίο είναι η αντιστοίχιση σημείων & η γεωμετρική επαλήθευση.

Πλάνο

Δεδομένου ότι εργαζόμαστε ροή βίντεο της τάξεως των 30 καρέ το δευτερόλεπτο, γνωρίζουμε πως το ίδιο χαρακτηριστικό σημείο δεν είναι εφικτό να απέχει πάνω από μια συγκεκριμένη απόσταση εικονοστοιχείων μεταξύ 2 συνεχόμενων καρέ

Βασιζόμενοι σε αυτήν την παραδοχή θα επιχειρήσουμε να αντιστοιχήσουμε τα χαρακτηριστικά σημεία με την καλύτερη δυνατή προσέγγιση

ΕΡΩΤΗΣΕΙΣ;

“It’s not the answer that enlightens but the question”

- Eugene Ionesco

ΕΥΧΑΡΙΣΤΟΥΜΕ!

Εκτιμούμε την παρουσία και το ενδιαφέρον σας βαθύτατα :)

- Παρουσίαση Παντελεήμων Σταματάκης (pastamatakis@gmail.com)
- Επιτήρηση Ιωάννης Βουρβουλάκης (jvourv@ihu.gr)
- Διπλ. εργασία Επιτάχυνση εξαγωγής χαρακτηριστικών εικόνας ORB με χρήση FPGA
- Για το Μεταπτυχιακό Ρομποτικής (2020-2021)
- Στο Διεθνές Πανεπιστήμιο της Ελλάδος