ΑΝΩΤΑΤΟ
ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ROBOT vs HUMAN
INTELLIGENT SYSTEM
PLAYING BOARD GAMES

Καράμπελας Δημήτρης
Σάσαρης Κωνσταντίνος
Χαραλαμπάκης Κωνσταντίνος

Επιβλέπων Καθηγητής: Υπ. Διδ. Σκουνάκης Εμμανουήλ M.Sc, M.Sc.

Χανιά 2010

ΕΥΧΑΡΙΣΤΙΕΣ

Στο σημείο αυτό νιώθουμε την ανάγκη να ευχαριστήσουμε από τα βάθη της καρδιάς μας τον καθηγήτη μας, κύριο Σκουνάκη Εμμανουήλ, για την αμέριστη συμπαράσταση και την σημαντική βοηθειά που μας πρόσφερε όλο αυτό το διάστημα,για την στηριξή του και την αδιάκοπη μετάδοση των πολύτιμων γνώσεων του, στον τομέα της Επεξεργασίας Εικόνας.

Ευχαριστούμε θερμά τον φίλο και συνάδελφο Σακάρο Θωμά για την παραχώρηση του ρομποτικού βραχίονα και την σημαντική βοηθειά του στην σύνδεση του ρομποτικού βραχίονα με το συστημά μας.

Ευχαριστούμε επίσης τους καθηγητές, κύριο Φουσκιτάκη Γεώργιο και κύριο Δοιτσίδη Ελευθέριο για την βοήθεια τους στην λύση του αντίστροφου κινηματικού προβλήματος για την κίνηση του βραχίονα.

ΠΕΡΙΛΗΨΗ

Η ραγδαία εξέλιξη των υπολογιστών τις τελευταίες δεκαετίες επέτρεψαν την ανάπτυξη ενός επιστημονικού κλάδου ο οποίος είναι γνωστός ως επεξεργασία εικόνας (Ψ.Ε.Ε). Η επεξεργασία εικόνας πλέον αποτελεί ολόκληρη επιστήμη, με πολλά ερευνητικά προγράμματα να γίνονται ανά τον κόσμο αφού είναι χιλιάδες οι εφαρμογές που διευκολύνουν την καθημερινοτητά μας, όπως για παράδειγμα, την αυτοματοποίηση γραφείου, την ρομποτική και την όραση μηχανής (computer vision), στρατιωτικές εφαρμογές, ιατρικα μηχανήματα, κλπ. Με την λέξη εικόνα δεν εννοούμε απλά την απεικόνιση μιας σκηνής, αλλά ένα μέσο με το οποίο μπορούμε να αποτυπώσουμε διάφορες πληροφορίες. Έτσι έγγραφα, ιατρικά δεδομένα όπως (υπερηχογραφήματα, μαγνητικές τομογραφίες) διαστημικά δεδομένα κ.α. μπορούν να ψηφιοποιηθούν και να επεξεργασθούν ως εικόνες.

Το αντικείμενο της συγκεκριμένης πτυχιακής εργασίας είναι η δημιουργία ενός εντελώς αυτόνομου και έξυπνου συστήματος το οποίο είναι σε θέση να βλέπει, να κρίνει, να αποφασίζει και να εκτελεί μόνο του μια συγκεκριμένη εργασία. Σχεδιάστηκε και κατασκευάστηκε ένα σύστημα με το οποίο ο άνθρωπος μπορεί να παίζει το παιχνίδι της τρίλιζας με αντίπαλο έναν ρομποτικό βραχίονα τεσσάρων βαθμών ελευθερίας ο οποίος αποτελεί την πτυχιακή εργασία του συναδέλφου Σακάρου Θωμά. Μέσω μιας κάμερας, ο βραχίονας βλέπει τον χώρο της σκακιέρας και με μία σειρά αλγορίθμων αναγνωρίζει τον χώρο, τα πιόνια, αλλα και την ακριβή θέση των πιονιών μέσα στον χώρο της σκακιέρας. Έπειτα γράφτηκε ένας άλλος αλγόριθμος με τον οποίο ο βραχίονας αναγνωρίζει και εκτελεί την πιο κατάλληλη και συμφέρουσα γι'αυτόν κίνηση ώστε να μπορέσει να νικήσει τον αντιπαλό του, που στην προκειμένη περίπτωση είναι ο άνθρωπος.

Το σύστημα μπορεί να το χειριστεί οποιοσδήποτε χωρίς να απαιτούνται εξειδικευμένες γνώσεις. Για τον λόγο αυτό δημιουργήθηκε ένα εικονικό περιβάλλον (front end environment), φιλικό σε κάθε χρήστη, με την γλώσσα προγραμματισμού Visual Basic 6.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 ΑΝΤΙΚΕΙΜΕΝΟ ΤΗΣ ΕΡΓΑΣΙΑΣ

Η πτυχιακή εργασία ασχολείται με έναν από τούς πιο ραγδαία εξελισσόμενους επιστημονικούς κλάδους σε παγκόσμιο επίπεδο, αυτόν της Επεξεργασίας εικόνας (Image Processing). Έχει κατασκευαστεί ένα πλήρως αυτόνομο και έξυπνο σύστημα, στο οποίο με επεξεργασία εικόνας να γίνεται δυνατή η αναγνώριση προτύπων, δηλαδή αντικειμένων μέσω μιας κάμερας σε έναν συγκεκριμένο χώρο που επιθυμούμε εμείς. Συγκεκριμένα στην εργασία αυτή τα αντικείμενα είναι τα πιόνια και ο χώρος είναι η σκακιέρα. Με την βοήθεια του κώδικα που έχει γραφτεί σε γλώσσα προγραμματισμού C γίνεται η λήψη αποφάσεων για τις κινήσεις μέσα από ένα σύνολο κανόνων για το επιτραπέζιο παιχνίδι της τρίλιζας. Οι κινήσεις του ¨αντιπάλου¨ γίνονται από έναν ρομποτικό βραχίονα τεσσάρων βαθμών ελευθερίας.

1.2 ΣΤΟΧΟΙ ΤΗΣ ΕΡΓΑΣΙΑΣ

- Εμβάθυνση στην επιστήμη της Ψηφιακής Επεξεργασίας Εικόνας.
- Εφαρμογή αλγορίθμων επεξεργασίας εικόνων.
- Εφαρμογή επεξεργασίας εικόνας σε ρομποτικά συστήματα

ΚΕΦΑΛΑΙΟ 2

ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

2.1 ΕΙΣΑΓΩΓΗ

Η περιοχή της ψηφιακής επεξεργασίας και ανάλυσης εικόνας έχει αναπτυχτεί δραστικά τις τελευταίες δεκαετίες. Σ' αυτο συνέβαλε τόσο η εξέλιξη των υπολογιστών όσο και η ανάπτυξη νέων επιστημονικών περιοχών. Τα θέματα που καλύπτει η ψηφιακή επεξεργασίας και ανάλυσης εικόνας είναι καθαρά τεχνολογικά και πλήρως εφαρμοσμένα. Έτσι παράλληλα με την εξέλιξη των υπολογιστών, έχουν αναπτυχτεί πολλές νέες εφαρμογές που αφορούν θέματα όπως η ψηφιακή επεξεργασία εγγράφων, η ρομποτική όραση, η βιοϊατρικη, ο βιομηχανικός ποιοτικός έλεγχος, τα πολυμέσα, κα

2.2 ΨΗΦΙΑΚΗ ΕΙΚΟΝΑ

Ψηφιακή εικόνα ονομάζουμε την μετάβαση από τον πραγματικό κόσμο (αναλογικό σήμα) στον Η/Υ (ψηφιακό σήμα). Έτσι μια πραγματική εικόνα μεταφέρεται στον ψηφιακό κόσμο με την μορφή διακριτού σήματος που έχει την μορφή ψηφιακών πινάκων. Μια ψηφιακή εικόνα μπορεί να είναι δυαδική (binary image), μονοχρωματική (αποχρώσεων του γκρι - grayscale image) ή έγχρωμη (color image). Μια ψηφιακή εικόνα αποχρώσεων του γκρι διαστάσεων N×M παριστάνεται από έναν δισδιάστατο πίνακα ακέραιων αριθμών I(i , j), i = 1,….,N και j = 1,…., M , όπου $0 \leq I(i , j) \leq G-1$. Το G ισούται με μια δύναμη του 2, δηλαδή $G= 2^m$ με συνηθέστερη τιμή το m=8 που αντιστοιχεί σε 256 αποχρώσεις του γκρι. (δηλαδή η τιμή 0 αντιστοιχεί στο μαύρο και η τιμή 255 αντιστοιχεί στο άσπρο με ενδιάμεσες 255 αποχρώσεις του γκρι)

Η τιμή I(i , j) είναι ανάλογη της φωτεινότητας στο εικονοστοιχείο (pixel) i,j και συνεπώς ο πίνακας I(i , j) ειναι ουσιαστικά μια διακριτή συνάρτηση που εκφράζει την ένταση της φωτεινότητας της εικόνας σε κάθε εικονοστοιχείο.

Η απλούστερη μορφή μίας εικόνας είναι η δυαδική μορφή. Μία δυαδική εικόνα έχει μόνο δυο στάθμες φωτεινότητας που συνήθως είναι το μαύρο και το άσπρο. Το μαύρο αντιστοιχεί στην τιμή 0 και το άσπρο στην τιμή 1.

Οι έγχρωμες εικόνες ψηφιακές εικόνες αποτελούν το μέσο για την απεικόνιση του πραγματικού κόσμου. Μια έγχρωμη εικόνα αποτελείται από τρείς εικόνες. Δηλαδή, το χρώμα κάθε εικονοστοιχείου έχει τρείς συνιστώσες. Από μαθηματικής άποψης, μια ψηφιακή έγχρωμη εικόνα διαστάσεων N×M μπορεί να συμβολιστεί ως $I_c(i,j)$, i = 1,….,N και j = 1,…., M , όπου $0 \leq I_c (i , j) \leq G-1$ για κάθε c = 1, 2, 3. Έτσι το χρώμα κάθε εικονοστοιχείου (i , j) προκύπτει από τον συνδυασμό τριών διαφορετικών αποχρώσεων **:**

$$\textbf{Color } (i, j) = [I_1 (i, j), I_2 (i, j), I_3 (i, j)]$$

2.3 ΧΡΩΜΑΤΙΚΑ ΜΟΝΤΕΛΑ

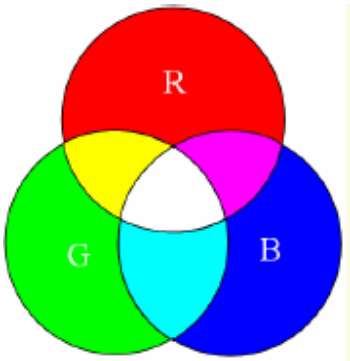
Ο τρόπος υπολογισμού του χρώματος κάθε εικονοστοιχείου εξαρτάται από το χρωματικό μοντέλο που θα χρησιμοποιήσουμε. Τα πιο γνωστά χρωματικά μοντέλα είναι τα εξής**:**

- ΠΡΟΣΘΕΤΙΚΟ (RG.B.)
- ΑΦΑΙΡΕΤΙΚΟ (C.M.Y.K)
- ΑΝΤΙΚΕΙΜΕΝΙΚΟ (H.S.B, Hue-Saturation-Brightness)

2.3.1 ΧΡΩΜΑΤΙΚΟ ΜΟΝΤΕΛΟ RGB

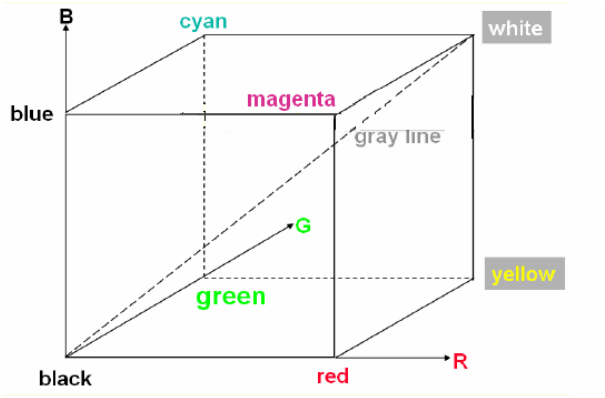
Η ονομασία του προέρχεται από τα αρχικά των λέξεων Red Green Blue (Κόκκινο Πράσινο Μπλε). Με τα βασικά αυτά χρώματα το μοντέλο κωδικοποιεί όλα τα χρώματα που μπορούν να εμφανιστούν σε μία οθόνη (συνήθως υπολογιστή). Στην μορφή του χρωματικού αυτού μοντέλου με βάθος χρώματος των 8 δυαδικών ψηφίων (bit) κάθε χρώμα μπορεί να παρασταθεί με μία τριάδα αριθμών και τιμές από 0 έως 255. Το μοντέλο βασίζεται στο γεγονός ότι όταν μία οθόνη δεν εκπέμπει φως εμφανίζεται μαύρη. Τα υπόλοιπα χρώματα δημιουργούνται με υπέρθεση των τριών βασικών με συγκεκριμένη αναλογία. Για το λόγο αυτό, το μοντέλο χαρακτηρίζεται και ως προσθετικό. Τα βασικά, τα δευτερογενή χρώματα και μερικά παραδείγματα δίνονται παρακάτω στην 8 bit έκδοση του μοντέλου:

- Μαύρο: (0,0,0)
- Λευκό: (255,255,255)
- Κόκκινο: (255,0,0)
- Πράσινο: (0,255,0)
- Μπλε: (0,0,255)
- Κίτρινο: (255,255,0)
- Γαλάζιο: (0,255,255)
- Ματζέντα (Magenta): (255,0,255)
- Πορτοκαλί: (255,102,0)



Σχήμα 2.3.1.1 Μοντέλο RGB

Το μοντέλο αυτό μπορεί να παρασταθεί με έναν κύβο χρωμάτων σε ένα καρτεσιανόο σύστημα αξόνων. Στην αρχή των αξόνων είναι η κορυφή του κύβου που αντιστοιχεί στο μαύρο χρώμα, ενώ στις κορυφές του κύβου που βρίσκονται πάνω στους άξονες βρίσκονται τα βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε). Τα δευτερογενή χρώματα βρίσκονται στις τρεις κορυφές του κύβου που βρίσκονται απέναντι από τα αντίστοιχα βασικά χρώματα και στην κορυφή απέναντι από το μαύρο βρίσκεται το λευκό. Κάθε χρώμα στο σύστημα αυτό προσδιορίζεται από ένα σημείο στον κύβο με τρεις συντεταγμένες. Στη διαγώνιο μεταξύ μαύρου και λευκού βρίσκονται όλες οι αποχρώσεις του γκρι. Στην 16 bit μορφή της μεθόδου τα δυνατά χρώματα είναι 2^16=65536 αντί 256 του 8 bit.



Σχήμα 2.3.1.2 Γραφική απεικόνιση του 8bit χρωματικού μοντέλου RGB

2.3.2 ΧΡΩΜΑΤΙΚΟ ΜΟΝΤΕΛΟ CMYK

Στην εκτύπωση των εντύπων χρησιμοποιείται ευρέως το σύστημα CMYK. Τα τρία βασικά χρώματα εδώ είναι: το Γαλάζιο (Cyan), το Ματζέντα (Magenta) & το Κίτρινο (Yellow).

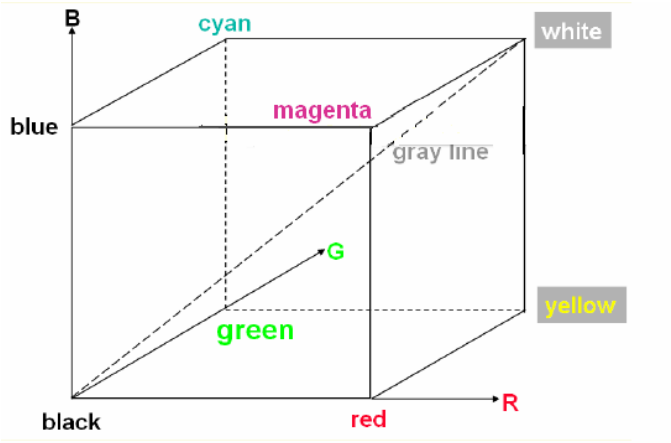
Με τα τρία αυτά χρώματα δημιουργούνται τα δευτερογενή Κόκκινο, Πράσινο & Μπλε ως εξής:

- Κόκκινο: Ματζέντα + Κίτρινο
- Πράσινο: Κίτρινο + Γαλάζιο
- Μπλε: Γαλάζιο + Ματζέντα

ΣΧΗΜΑ 2.3.2.1 Μοντέλο RGB

Το μοντέλο αυτό βασίζεται στο γεγονός ότι το υπόβαθρο της εκτύπωσης είναι το λευκό χαρτί που ανακλά όλα τα χρώματα (μήκη κύματος). Κάθε βασικό χρώμα που προστίθεται με ένα μελάνι απορροφά ορισμένα χρώματα και αποδίδει τα υπόλοιπα. Για παράδειγμα το κίτρινο μελάνι απορροφά το μπλε χρώμα και αφήνει το πράσινο και το κόκκινο να ανακλαστεί. Εδώ ο συνδυασμός των τριών βασικών χρωμάτων δίνει το μαύρο χρώμα (πλήρης απορρόφηση των ακτινοβολιών). Για το λόγο αυτό το μοντέλο CMY. χαρακτηρίζεται και ως "αφαιρετικό". Το Μαύρο χρώμα, επίσης, προκύπτει από το συνδυασμό ενός βασικού και του αντίθετου δευτερογενούς:

- Μαύρο: Γαλάζιο + Ματζέντα + Κίτρινο
- Μαύρο: Γαλάζιο + Κόκκινο
- Μαύρο: Ματζέντα + Πράσινο
- Μαύρο: Κίτρινο + Μπλε

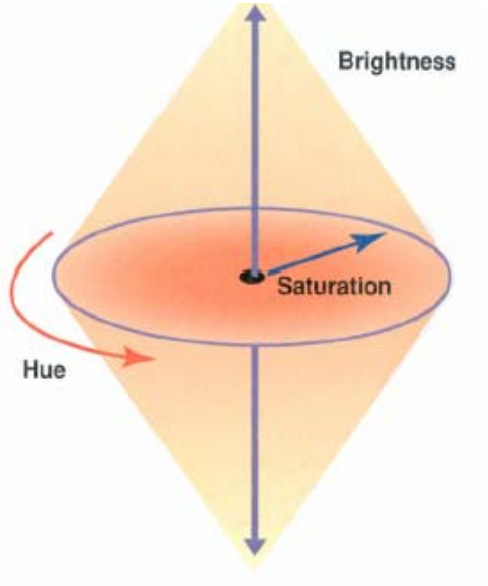


Σχήμα 2.3.2.2 Γραφική απεικόνιση του χρωματικού μοντέλου CMY.

Τα μελάνια, όμως, από τη φύση τους δεν μπορούν να αποδώσουν συγκεκριμένα μήκη κύματος – χρώματα (όπως τα εικονοστοιχεία (pixels) μίας οθόνης) αλλά, μία πιο ευρεία περιοχή του χρωματικού φάσματος. Το αποτέλεσμα είναι ο συνδυασμός των τριών βασικών χρωμάτων να δίνει ένα καφετί χρώμα αντί για το μαύρο. Για το λόγο αυτό προστέθηκε στο μοντέλο CMY και το μαύρο μελάνι, με αποτέλεσμα να προκύψει το χρωματικό μοντέλο CMYK (Cyan – Magenta – Yellow – Black). Πρακτικά στην εκτύπωση δεν χρησιμοποιείται σήμερα το CMY μοντέλο αλλά το CMYK. Το μοντέλο CMY μπορεί να παρασταθεί όπως και το RGB με ένα κύβο σε ένα καρτεσιανό σύστημα αξόνων με το λευκό χρώμα στην αρχή των αξόνων και τα βασικά χρώματα επάνω στους άξονες.

2.3.3 ΧΡΩΜΑΤΙΚΟ ΜΟΝΤΕΛΟ H.S.B

Με τις μεθοδολογίες HSB (Hue, Saturation, Brightness) μπορούμε να καθορίσουμε την απόχρωση (hue) με γωνιακούς όρους (από 0 έως 360 βαθμούς) και τις παραμέτρους κορεσμού (Saturation), φωτεινότητας (Brightness) ως ποσοστά επί τις %.



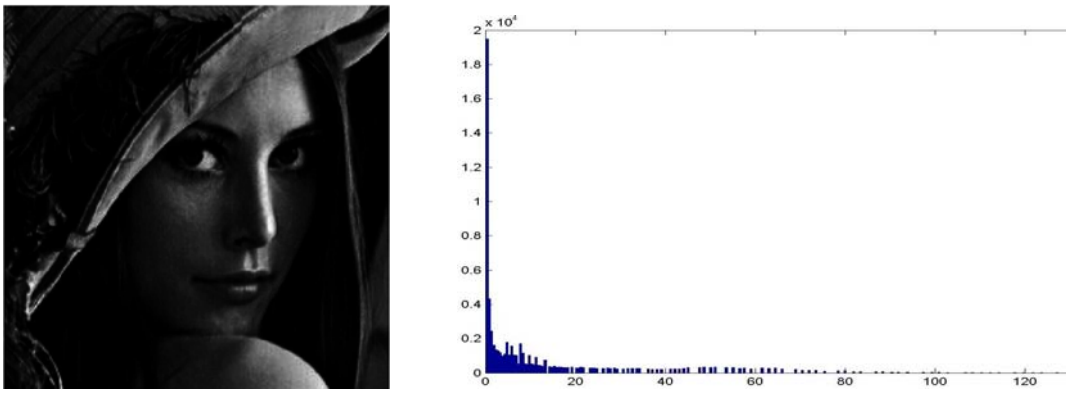
Σχήμα 2.3.3.1 Μοντέλο HSB

Οι παράμετροι φωτεινότητας (Brightness) σκίασης αντιστοιχούν στο ποσοστό μίξης του μαύρου ή λευκού με το χρώμα. Ποσοστό 100% στη σκίαση αντιστοιχεί στο λευκό χρώμα ενώ 0% αντιστοιχεί στο μαύρο. Το καθαρό χρώμα αντιστοιχεί σε ποσοστό 50% σκίασης. Τα αντίστροφα ποσοστά ισχύουν και στην παράμετρο φωτεινότητας.

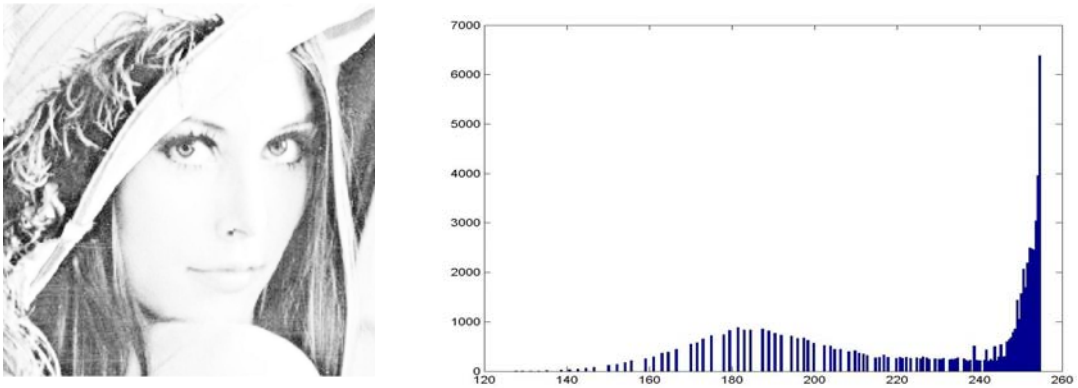
Ο κορεσμός (Saturation) είναι η ένταση του χρώματος. Ποσοστό 100% κορεσμού αντιστοιχεί στο καθαρό χρώμα. Ποσοστό 0% αντιστοιχεί σε άσπρο, μαύρο ή γκρι χρώμα ανάλογα με την απόχρωση (hue).

2.4 ΙΣΤΟΓΡΑΜΜΑ

Το ιστόγραμμα είναι μια ένδειξη της στοχαστικής κατανομής της εικόνας. Το ιστόγραμμα μιας εικόνας αποχρώσεων του γκρι περιέχει σημαντικές πληροφορίες για την εικόνα και για τον λόγο αυτό είναι ένα από τα σημαντικότερα εργαλεία στην επεξεργασία ψηφιακών εικόνων. Μπορεί να χρησιμοποιηθεί για την βελτιστοποίηση της εικόνας, την τροποποίηση των οπτικών χαρακτηριστικών της, την εξαγωγή χαρακτηριστικών της εικόνας κ.α.



Εικόνα 2.4.1 Παράδειγμα ιστογράμματος σκοτεινής εικόνας



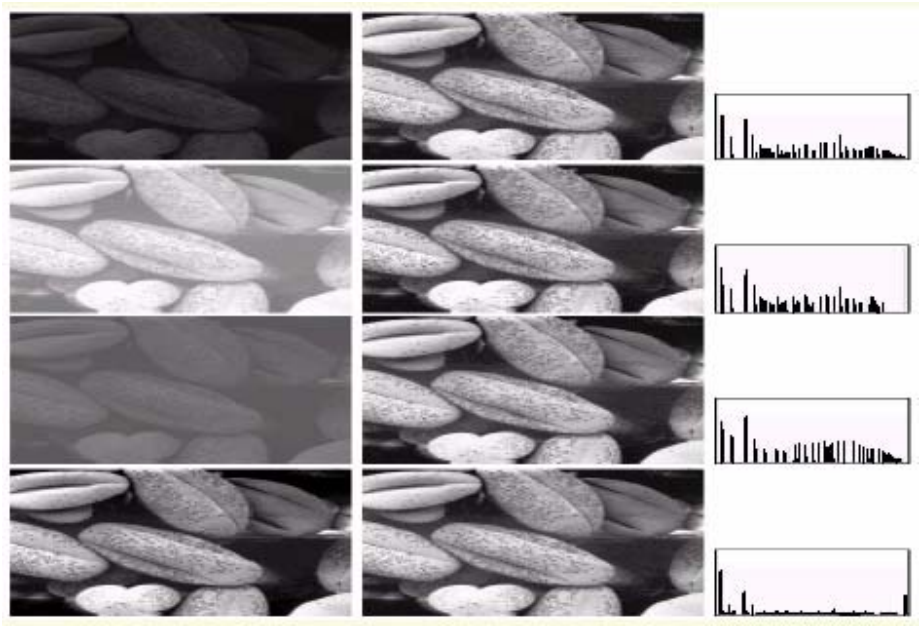
Εικόνα 2.4.2Παραδειγμα ιστογράμματος φωτεινής εικόνας

2.4.1 ΕΞΙΣΟΡΟΠΗΣΗ ΙΣΤΟΓΡΑΜΜΑΤΟΣ

Εξισορρόπηση ιστογράμματος είναι η διαδικασία κατά την οποία κατανέμουμε τα διακριτά επίπεδα του γκρι μιας εικόνας ομοιόμορφα σε όλη την διαθέσιμη δυναμική περιοχή τιμών. Είναι μια χρήσιμη διαδικασία για την βελτίωση της εικόνας, για την συμπίεση της και για την κατάτμηση της.

Η παρακάτω μαθηματική έκφραση μας δίνει μια εκτίμηση της πιθανότητας εμφάνισης του r_k επιπέδου του γκρι στην εικόνα **:**

$$\mathbf{h(r_k)= n_k/n}$$



Εικόνα 2.4.1.1 Ιστογράμματα

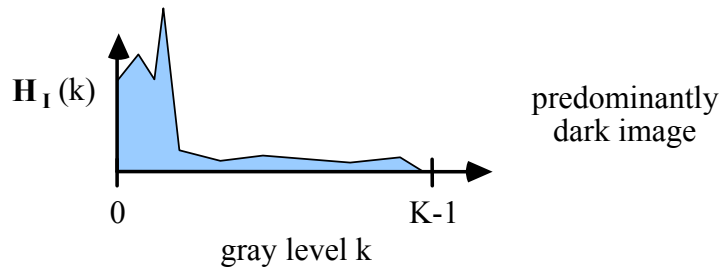
Μετά την εφαρμογή της εξισορρόπησης, το ιστόγραμμα εκτείνεται σε όλη την δυναμική περιοχή τιμών (μέχρι και τις πιο φωτεινές τιμές της κλίμακας) και έτσι εξηγείται το <<ξεθώριασμα>> της εικόνας.

$$S_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n}$$

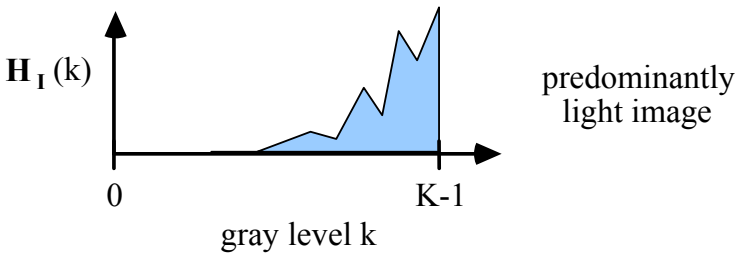
Σχέση 2.4.1.1

Όπου $s_κ$ η αντίστοιχη τιμή του n_k στο ισοσταθμισμένο ιστόγραμμα, n_j ο αριθμός εμφάνισης των pixels με τιμή r_k και n το σύνολο των pixels της εικόνας.

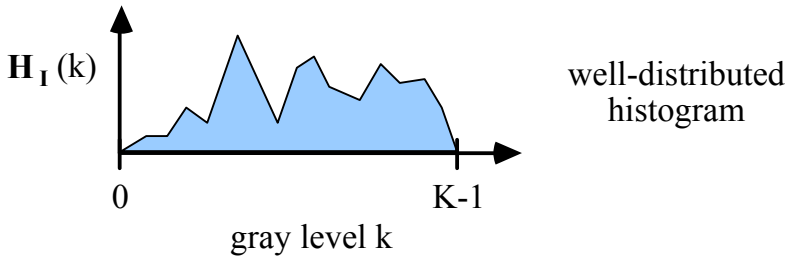
Τα παρακάτω ιστογράμματα δείχνουν (2.4.1.2)μια σκοτεινή εικόνα και (2.4.1.3) μια φωτεινή εικόνα, όπως μπορούμε να δούμε στις εικόνες αυτές οι περισσότερες τιμές είναι κοντά στο 0 για την πρώτη περίπτωση και κοντά στο 1 για την δεύτερη αυτό κάνει πιο δύσκολο το να ορίσουμε κατώφλι. Όπως βλέπουμε στο (2.4.1.4).ιστόγραμμα έχουμε καλύτερη χρήση της περιοχής της γκρι-κλίμακας πεδίων φωτεινότητας



Σχήμα 2.4.2.1 Ιστόγραμμα σκοτεινής εικόνας



Σχήμα 2.4.2.2 Ιστόγραμμα φωτεινής εικόνας

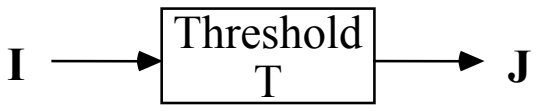


Σχήμα 2.4.2.3 Ιστόγραμμα εικόνας καλής κατανομής

2.4.2 ΚΑΤΩΦΛΙΩΣΗ

Κατωφλίωση ονομάζουμε την διαδικασία κατά την οποία επιλεγούμε έναν ακέραιο αριθμό στην περιοχή της γκρι-κλίμακας των επιπέδων φωτεινότητας της εικόνας ο οποίος ονομάζεται κατώφλι T, με σκοπό τον διαχωρισμό της σημαντικής από την μη σημαντική πληροφορία. Ας υποθέσουμε ότι μια γκρι-επιπέδων εικόνα I έχει K γκρι-επίπεδα φωτεινοτητας:0,1,2,....,K-1. Επιλέγουμε το επιθυμητό κατώφλι T το οποίο ανήκει {0, 1, 2 ,..., K-1 } και συγκρίνουμε κάθε επίπεδο φωτεινότητας στη γκρι-επιπέδων εικόνα I με το T(κατώφλι) αυτό που προκύπτει είναι μια δυαδική εικόνα J η οποία ορίζεται ως ακολούθως:

$$J(i, j) = '0' \text{ εάν } I(i, j) \geq T$$
$$J(i, j) = '1' \text{ εάν } I(i, j) < T$$



Η κατωφλιωση είναι η πιο απλή διαδικασία στην επεξεργασία εικόνας. Η δυαδική εικόνα J που παίρνουμε από την κατωφλίωση της εικόνας I**,** εξαρτάται πάρα πολύ από το κατώφλι T**.** Δεν φτάνει όμως μόνο αυτό για να πάρουμε την επιθυμητή δυαδική εικόνα. θα πρέπει το ιστόγραμμα στο οποίο ορίσαμε το κατώφλι να έχει σωστή κατανομή ώστε οι τιμές τους ιστογράμματος να είναι όσο πιο ξεκάθαρες γίνεται.

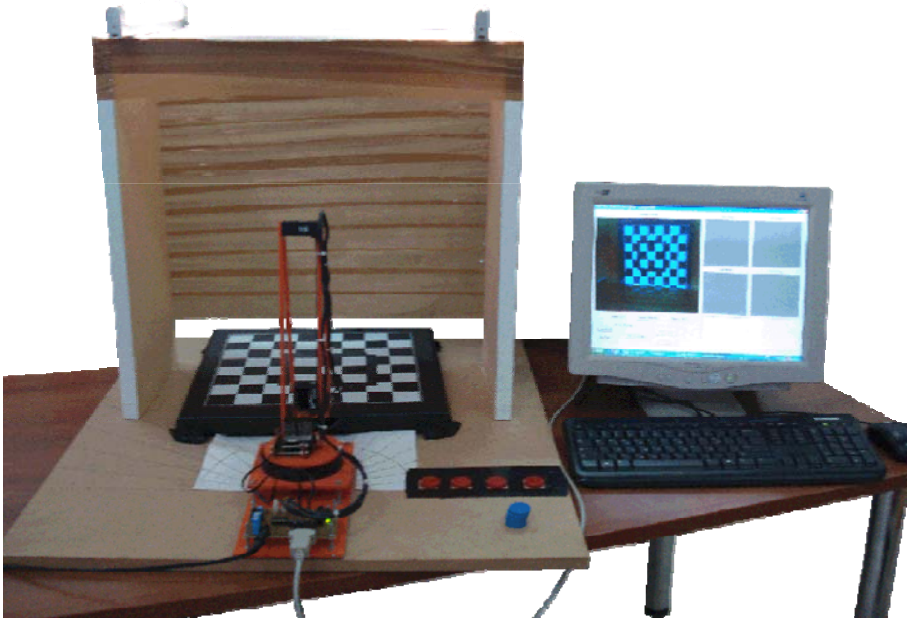
ΚΕΦΑΛΑΙΟ 3

ΑΝΑΛΥΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

3.1 ΕΙΣΑΓΩΓΗ

Το προτεινόμενο σύστημα που φαίνεται παρακάτω, είναι ένα έξυπνο αυτόματο σύστημα. Τα βασικά μηχανήματα που συνεργάζονται είναι:

- Ένας ρομποτικός βραχίονα
- Μια web camera υψηλής ανάλυσης
- Ένας υπολογιστής



Εικόνα 3.1.1Ρομποτικό Σύστημα

3.2 ΚΙΝΗΣΗ

Ο ρομποτικός βραχίονας ουσιαστικά είναι ο αντίπαλος στο παιχνίδι ο οποίος δέχεται εντολές μέσα από το πρόγραμμα (π.χ. από πού και πως θα πάρει το πιόνι, που και πως θα το αφήσει, κλπ).

Η κατασκευή και ρύθμιση του ρομποτικού βραχίονα έγινε στα πλαίσια της πτυχιακής εργασίας του συνάδελφου Σακάρου Θωμά. Όλες τις απαραίτητες πληροφορίες για την σχεδίαση, την υλοποίηση αλλά και τον προγραμματισμό του ρομποτικού βραχίονα θα τις βρείτε στο σύγγραμμα της πτυχιακής εργασίας του με τίτλο «Κατασκευή και έλεγχος αρθρωτού ρομποτικού βραχίονα τεσσάρων βαθμών ελευθερίας»

Εικόνα 3.2.1Ρομποτικός βραχίονας

3.3 ΜΗΧΑΝΙΚΗ ΟΡΑΣΗ

Η κάμερα που χρησιμοποιήθηκε είναι η QUICKCAM Communicate STX της εταιρείας Logitech. Βασίζεται στην τεχνολογία CMOS με αισθητήρα VGA, έχει μέγιστη ανάλυση 1.3 Megapixels μέσω software, έως 30 fps, σύνδεση USB 2.0.

Εικόνα 3.3.1Web camera

Κατά τη δημιουργία της πτυχιακής αυτής εργασίας, τα στάδια υλοποίησης ήταν:

Η κατασκευή και εκτέλεση των παρακάτω τριών εκδόσεων των προγραμμάτων υποστήριξης του συστήματος:

- Manual version
- Automatic version
- Autonomous automatic version με εικονικό περιβάλλον.

3.4 MANUAL VERSION

Δημιουργήθηκε το εκτελέσιμο αρχείο TicTacToe_CSK_190510.exe με την γλώσσα προγραμματισμού C ο κώδικας (παράρτημα) του οποίου ενημερώνει τα παρακάτω txt αρχεία και περιέχει την έξυπνη λογική (βλ. 3.6.3 Λήψη αποφάσεων) του συστήματος ώστε να είναι σε θέση ο ρομποτικός βραχίονας να κρίνει και να επιλέγει την πιο κατάλληλη γι'αυτόν κίνηση.

Στην χειροκίνητη λειτουργία, πρέπει ο χρήστης να δηλώσει κάποιες παραμέτρους στο πρόγραμμα. Οι παράμετροι αυτοί δηλώνονται σε εξωτερικά αρχεία .txt, τα οποία δημιουργούνται και αλλάζουν χειροκίνητα από τον χρήστη στον C:\. Τα απαραίτητα .txt αρχεία είναι τα εξής:

- Player.txt
- Positionh.txt
- Robotm.txt
- Pinakas.txt
- Kinisi.txt
- Winner.txt

3.4.1 Player.txt

Στο αρχείο Player.txt ορίζουμε τον παίκτη που θα ξεκινήσει πρώτος την παρτίδα της τρίλιζας. Ο αριθμός ΄΄1΄΄ για τον άνθρωπο και ο αριθμός ΄΄2΄΄ για τον ρομποτικό βραχίονα.

3.4.2 Positionh.txt

Στο αρχείο Positionh.txt ορίζουμε την θέση που επιλέγει ο άνθρωπος να παίξει στην σκακιέρα σε μία από τις εννέα διαθέσιμες θέσεις οι οποίες αντιστοιχούν στις τιμές ΄΄1΄΄ -΄΄9΄΄.

3.4.3 Robotm.txt

Στο αρχείο Robotm.txt αποθηκεύεται η θέση που επέλεξε ο ρομποτικός βραχίονας να τοποθετήσει το πιόνι του.

3.4.4 Pinakas.txt

Στο αρχείο Pinakas.txt κάθε φορά που επιλέγετε μία κίνηση από τον άνθρωπο ή από τον ρομποτικό βραχίονα ενημερώνεται ένας πίνακας εννέα θέσεων με τον αριθμό ΄΄1΄΄ για τον πρώτο και με τον αριθμό ΄΄2΄΄ για τον δεύτερο, για κάθε θέση που αντιστοιχεί η κίνηση αυτή. Ώστε να είναι σε θέση ο ρομποτικός βραχίονας να γνωρίζει ποιες θέσεις είναι κατειλημμένες και ποιες είναι ελεύθερες. (π.χ. 001001002)

3.4.5 Kinisi.txt

Στο αρχείο Kinisi.txt μας ενημερώνει το πρόγραμμα για την γραμμή (i) και την στήλη (j) που βρίσκεται το κεντρικό pixel (βλ. 3.6.2) του πιονιού.

3.4.6 Winner.txt

Στο αρχείο Winner.txt μας ενημερώνει το πρόγραμμα ποιος είναι ο νικητής της παρτίδας με τον αριθμό ΄΄1΄΄ για τον άνθρωπο και τον αριθμό ΄΄2΄΄ για τον ρομποτικό βραχίονα.

Όλα τα .txt αρχεία πρέπει να είναι άδεια πριν την εκκίνηση της παρτίδας εκτός:

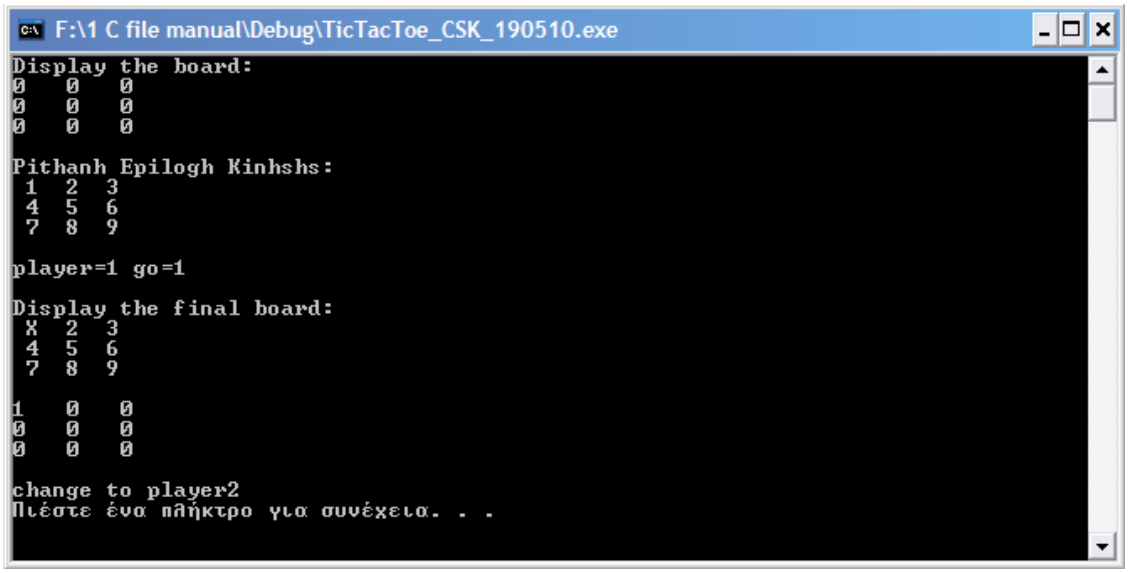
1. Από το αρχείο Winner.txt το οποίο πρέπει να έχει την τιμή ΄΄0΄΄.
2. Από το αρχείο Player.txt που πρέπει να έχει τιμή “1”.
3. Από το αρχείο Positionh.txt που πρέπει να έχει την τιμή της θέσης (1-9) που θα παίξει ο άνθρωπος.

3.4.7 ΔΙΑΔΙΚΑΣΙΑ MANUAL VERSION

Η διαδικασία που ακολουθείται στην χειροκίνητη έκδοση είναι η εξής:

Στην αρχή της διαδικασίας πρέπει να δημιουργήσουμε τα απαραίτητα .txt αρχεία στον C:\. Αφού τοποθετήσουμε στο player.txt την τιμή 1 για να ξεκινήσει πρώτος ο άνθρωπος και στο positionh.txt την τιμή 1 για να τοποθετήσει ο άνθρωπος στην θέση 1 της σκακιέρας πιόνι, τρέχουμε το αρχείο TicTacToe_CSK_190510.exe.

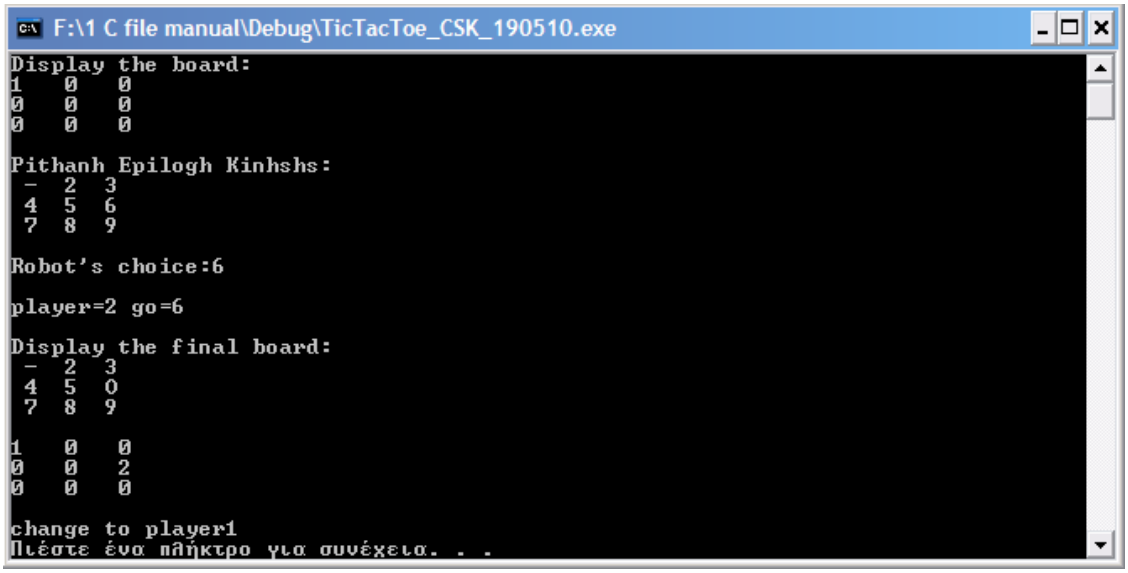
Εμφανίζεται το παρακάτω παράθυρο.



Εικόνα 3.4.7.1

Το παράθυρο αυτό μας πληροφορεί για τον αρχικό πίνακα που είναι άδειος, για τις διαθέσιμες θέσεις του που στην αρχή είναι εννέα, για την σειρά του παίκτη και την θέση που επέλεξε. Στην προκειμένη περίπτωση είναι ο παίκτης ένα και επέλεξε την θέση 1, έπειτα μας εμφανίζει τον πίνακα με τον αριθμό 1 στην θέση 1 για να μας δείξει ότι αυτή η θέση καλύφθηκε πια από τον παίκτη 1 που είναι ο άνθρωπος. Ενημερώνει τα txt αρχεία ώστε να γνωρίζει το robot ποια κίνηση έχει γίνει.

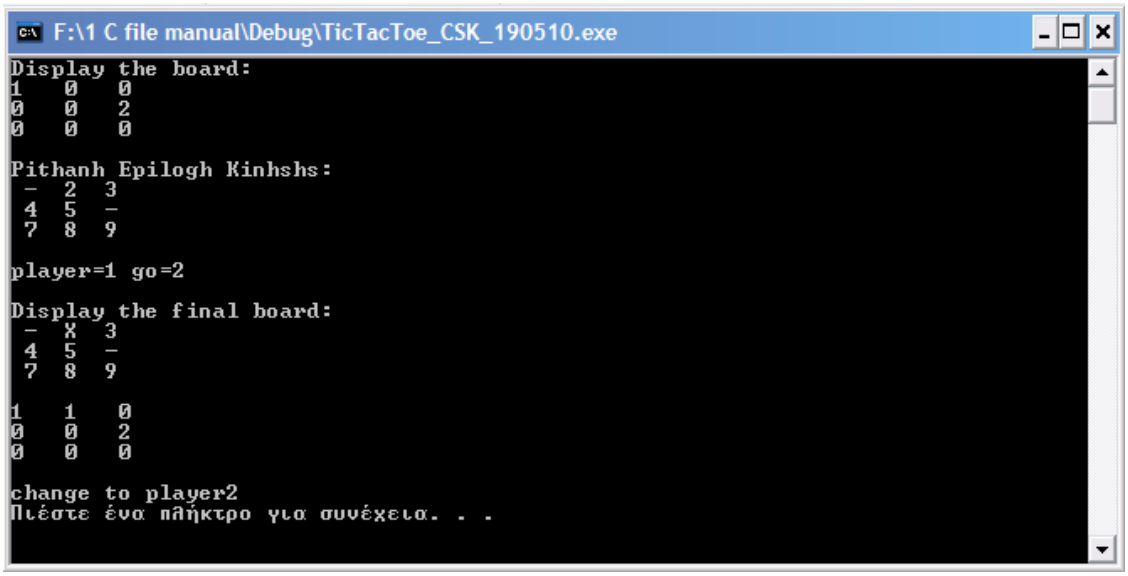
Τρέχουμε ξανά το εκτελέσιμο αρχείο TicTacToe_CSK_190510.exe και μας εμφανίζει το παράθυρο ενημερώνοντας μας για την κίνηση που επέλεξε το robot. Όπως φαίνεται, σε αυτή την περίπτωση ο ρομποτικός βραχίονας επέλεξε να βάλει πιόνι στην θέση 6. Έτσι εμφανίζεται ο πίνακας με την κίνηση του ανθρώπου αλλά και με την κίνηση του robot ενώ ταυτόχρονα μας δείχνει ποιες θέσεις παραμένουν ελεύθερες.



Εικόνα 3.4.7.2

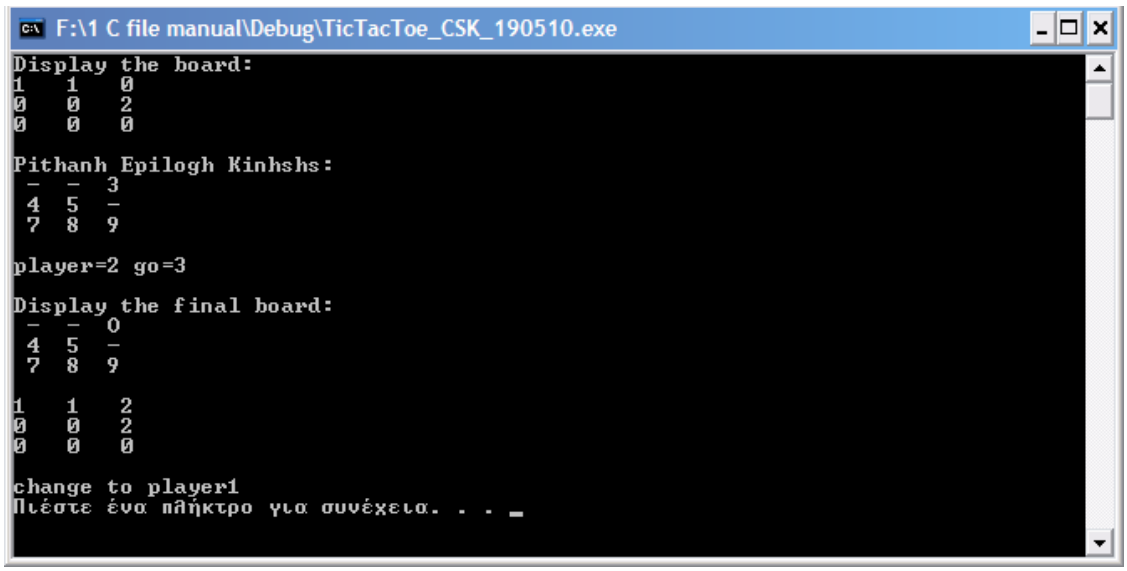
Τώρα είναι η σειρά του ανθρώπου. Ανοίγουμε το potition.txt, βάζουμε την τιμή εκείνη που αντιστοιχεί στην θέση που θέλουμε να βάλουμε πιόνι και τρέχουμε ξανά το αρχείο TicTacToe_CSK_190510.exe.

Όπως βλέπουμε στο παρακάτω παράθυρο η θέση αυτή είναι το κουτί 2.



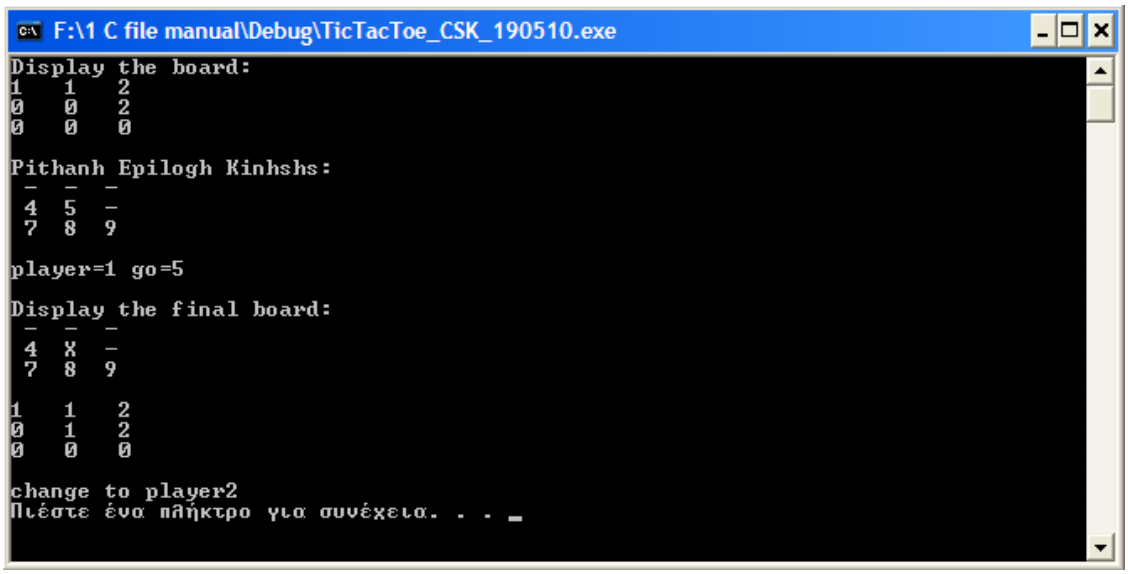
Εικόνα 3.4.7.3

Τρέχουμε ξανά το αρχείο TicTacToe_CSK_190510.exe, το robot επιλέγει την θέση 3 και μας εμφανίζει στο παράθυρο τις θέσεις που έχουν καλυφθεί αλλά και τις διαθέσιμες θέσεις του πίνακα.



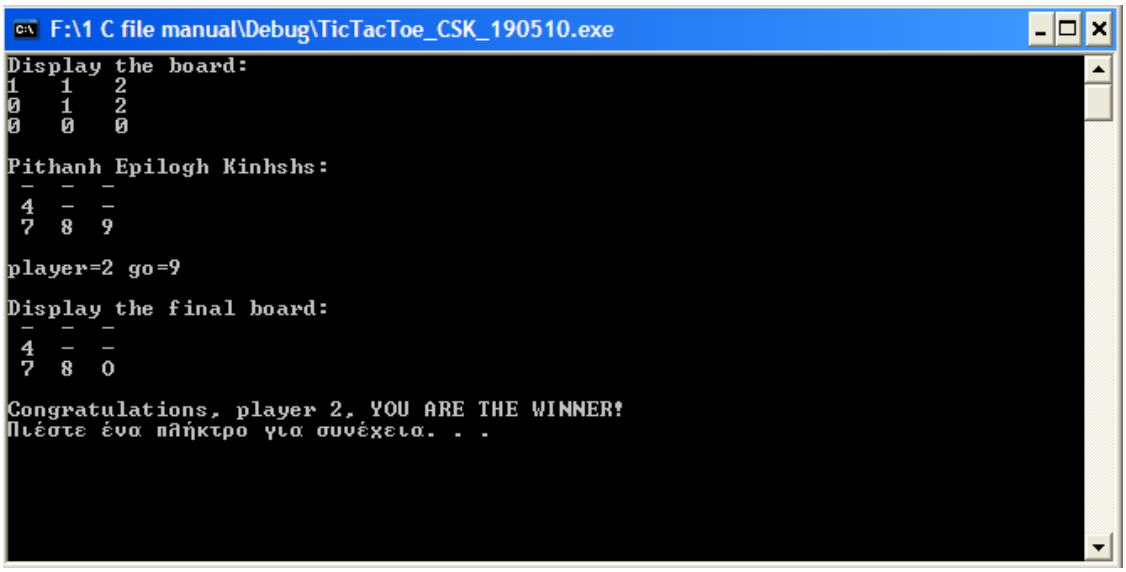
Εικόνα 3.4.7.4

Ανοίγουμε το αρχείο potition.txt βάζουμε την τιμή εκείνη που αντιστοιχεί στην θέση του πίνακα που επιθυμούμαι να βάλουμε το πιόνι τρέχουμε πάλι το αρχείο TicTacToe_CSK_190510.exe και μας εμφανίζει τον πίνακα με τις διαθέσιμες πια θέσεις



Εικόνα 3.4.7.5

Στο παρακάτω παράθυρο βλέπουμε την κίνηση του βραχίονα στην θέση 9 με την οποία εμπόδισε την νίκη του ανθρώπου , η επόμενη κίνηση του ανθρώπου στην θέση 8 σηματοδοτεί και την νίκη του.



Εικόνα 3.4.7.6

3.5 AUTOMATIC VERSION

Η έκδοση αυτή δημιουργήθηκε για να μπορούμε να παίρνουμε όλες τις απαραίτητες πληροφορίες όπως, ποιός παίκτης έχει σειρά, σε ποια θέση τοποθέτησε πιόνι ο άνθρωπος, σε ποια θέση τοποθέτησε πιόνι ο βραχίονας, ποιες θέσεις έχουν πιόνια από ποιον αλλά και ποιες θέσεις είναι ελεύθερες, αν υπάρχει νίκη και από ποιόν , από τα txt αρχεία χωρίς να τα βλέπουμε από το παράθυρο διαλόγου της C.

Έτσι η παραπάνω παρτίδα έχει ως εξής:

Player.txt : 1

Positionh.txt : 1

RUN TicTacToe_CSK_190510_auto.exe

Player.txt : 2

Robotm.txt:

Pinakas.txt:100000000

Kinisi.txt:

Winner.txt:0

RUN TicTacToe_CSK_190510_auto.exe

Player.txt : 1

Positionh.txt : 1

Robotm.txt:6

Pinakas.txt:100002000

Kinisi.txt:

Winner.txt:0

Positionh.txt : 2

RUN TicTacToe_CSK_190510_auto.exe

Player.txt : 2

Pinakas.txt:110002000

Kinisi.txt:

RUN TicTacToe_CSK_190510_auto.exe

Player.txt : 1

Robotm.txt:3

Pinakas.txt:112002000

Kinisi.txt:

Positionh.txt : 5

RUN TicTacToe_CSK_190510_auto.exe

Player.txt :2

Pinakas.txt:112012000

Kinisi.txt:

RUN TicTacToe_CSK_190510_auto.exe

Robotm.txt:3

Pinakas.txt:112002000

Kinisi.txt:

Winner.txt:1 (Νίκη ανθρώπου)

3.6 AUTOMATIC VERSION ΜΕ FRONT END ΠΕΡΙΒΑΛΛΟΝ.

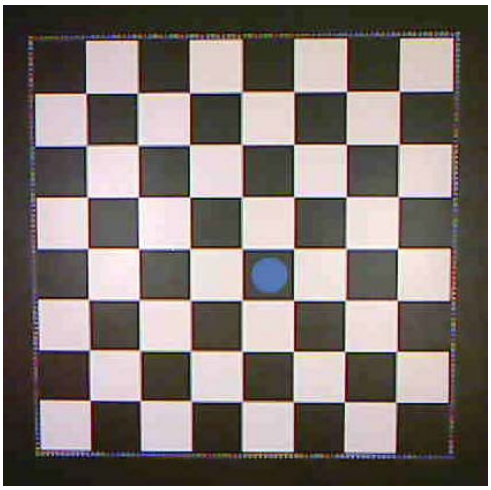
Η τελική έκδοση του συστήματος η οποία είναι εντελώς αυτόνομη χωρίς ο άνθρωπος να χρειάζεται να παρέμβει σε κανένα σημείο της διαδικασίας ήταν το επόμενο βήμα. Όλα τα απαραίτητα αρχεία txt ενημερώνονται αυτόματα από το FRONT END (ΕΙΚΟΝΙΚΟ ΠΕΡΙΒΑΛΛΟΝ 3.7)

Η κάμερα στην ουσία δίνει την δυνατότητα μηχανικής όρασης στο σύστημα . Μ' αυτό τον τρόπο μπορεί να αντιλαμβάνεται τις όποιες αλλαγές στον χώρο, που έχουν οριστεί. Στην προκειμένη περίπτωση ο χώρος αυτός είναι μια περιοχή εννέα θέσεων στην σκακιέρα.

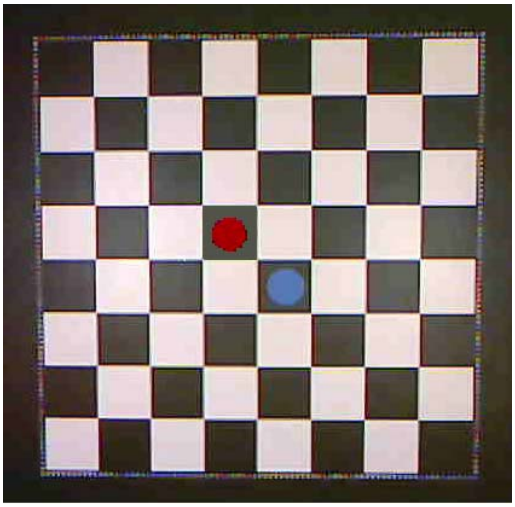
3.6.1 ΕΝΤΟΠΙΣΜΟΣ ΑΛΛΑΓΗΣ ΣΤΟΝ ΧΩΡΟ

Η κάμερα είναι προγραμματισμένη να τραβάει φωτογραφίες την περιοχή της σκακιέρας ανά 30 δευτερόλεπτα. Η πρώτη φωτογραφία αποθηκεύεται με την ονομασία prototype.bmp και εμφανίζει την σκακιέρα χωρίς κανένα πιόνι πάνω σε αυτή, μόλις περάσει το χρονικό διάστημα που του έχουμε ορίσει η κάμερα τραβάει την επόμενη εικόνα η οποία αποθηκεύεται ως new.bmp. Οι δύο αυτές εικόνες μετατρέπονται από τον κώδικα που έχουμε γράψει σε εικόνες .raw ώστε να είναι έτοιμες προς επεξεργασία.

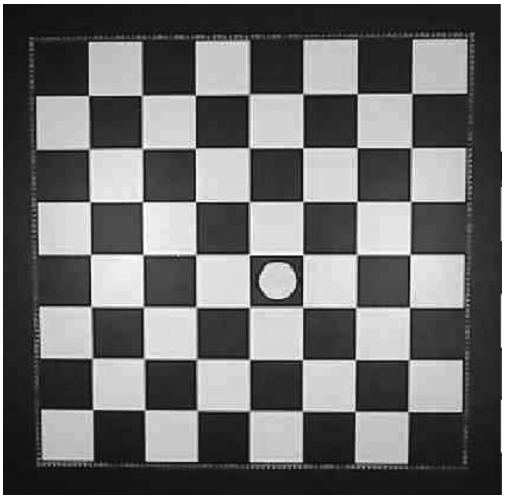
Σ' αυτό το σημείο αφαιρούμε όλα τα pixels της εικόνας prototype.raw από όλα τα pixels της εικόνας new.raw. Η διαφορά των δύο εικόνων δημιρουγεί μια εικόνα με ονομασία result.raw. Η εικόνα result.raw απεικονίζει τις αλλαγές μεταξύ των δύο εικόνων. Η επιθυμητή εικόνα πρέπει να περιέχει μόνο ένα πιόνι. Στην επόμενη κίνηση η new.bmp αποθηκεύεται σαν old.bmp και η νέα φωτογραφία θα αποθηκευτεί σαν new.bmp αυτή η διαδικασία ακολουθείται για κάθε φωτογραφία που παίρνει η κάμερα, ενώ κάθε φορά το αποτέλεσμα της διαφοράς τους θα είναι η εικόνα result.bmp.



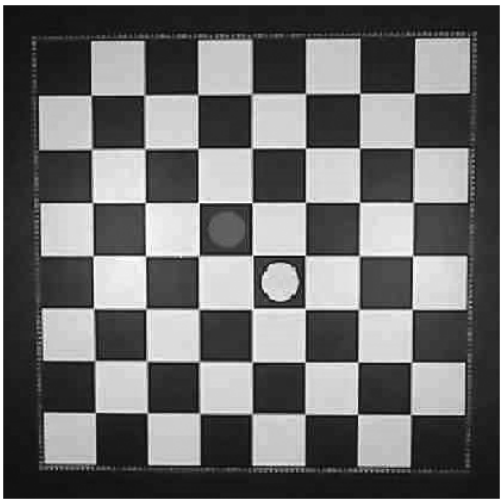
Εικόνα. 3.3.2.1 old .bmp



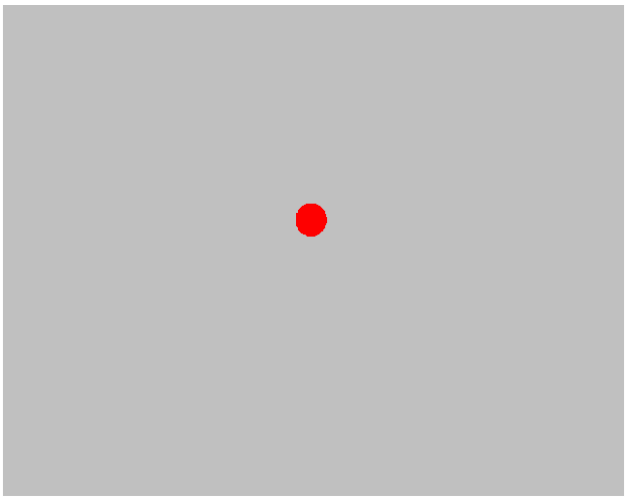
Εικόνα 3.3.2.2 new .bmp



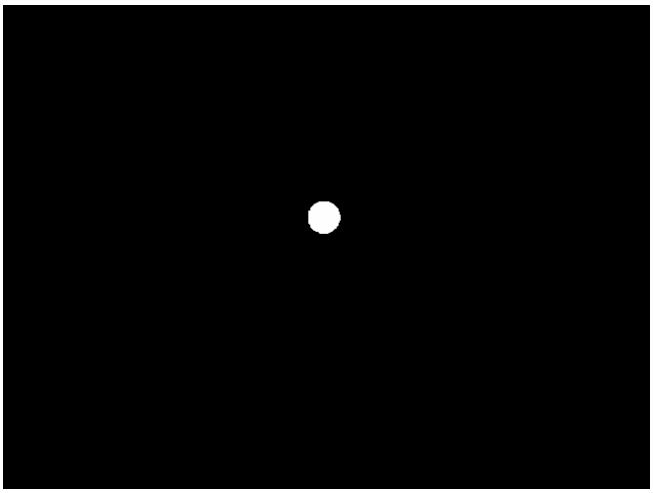
Εικόνα 3.3.2.3 grayscale old



Εικόνα 3.3.2.4 grayscale



Εικόνα 3.3.2.4 Διαφορά εικόνων .bmp

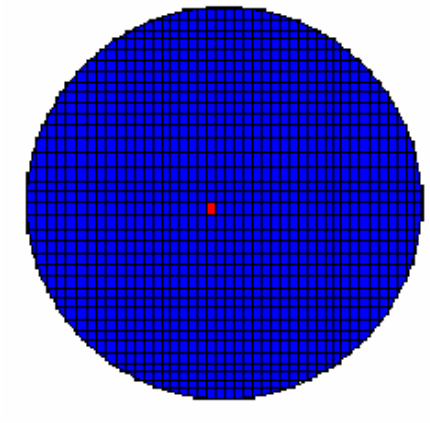


Εικόνα 3.3.2.4 Διαφορά εικόνων .raw

3.6.2 ΥΠΟΛΟΓΙΣΜΟΣ ΚΕΝΤΡΟΥ ΠΙΟΝΙΩΝ

Εδώ δημιουργήθηκε ένα πρόγραμμα (το Teliko_Center.exe) το οποίο κάθε φορά που είναι η σειρά του ανθρώπου να παίξει και αφού έχει βγει το result.raw εντοπίζει το κεντρικό pixel του πιονιού.

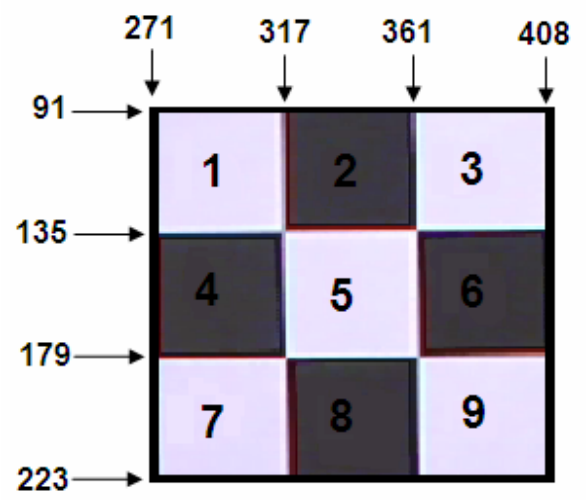
Διαβάζει την εικόνα result.raw, μετράει την πρώτη γραμμή του πίνακα που περιέχει τα pixels με την τιμή της grayscale τιμής που αντιστοιχεί στο χρώμα του πιονιού μας, κάθε φορά που βρίσκει στην σειρά ένα pixel με την τιμή αυτή, αυξάνεται ένας μετρητής κατά ένα, και αυτό γίνεται για όλες τις γραμμές του πίνακα..Όταν ο καταχωρητής φτάσει σε εκείνη την γραμμή στην οποία μέτρησε έναν συγκεκριμένο αριθμό από pixels στην σειρά με την τιμή που του έχουμε ορίσει τότε κάνει τον αριθμό αυτόν /2, και έτσι βρίσκει ακριβώς το κέντρο του πιονιού.



Εικόνα 3.6.2.1. Υπολογισμός κέντρου πιονιού

Αφού είμαστε σε θέση να γνωρίζουμε το κέντρο του πιονιού κατά i , j (γραμμή και στήλη της εικόνας) τότε μπορούμε να γνωρίζουμε και σε ποιο κουτί της σκακιέρας βρίσκεται το πιόνι αυτό. Ο αλγόριθμος που περιέχεται στο Teliko_Center.exe χωρίζει το κομμάτι της σκακιέρας που παίζουμε σε εννέα κουτιά, ονομάζοντας το κάθε κουτί με έναν αριθμό (από 1 έως 9). Διαβάζοντας την εικόνα έχουμε ορίσει στον κώδικα μας τον αριθμό που αντιστοιχεί σε κάθε γραμμή και σε κάθε στήλη της σκακιέρας όπως φαίνεται στην εικόνα (Εικόνα 4.6.2.2).

Αν π.χ. το κέντρο ενός πιονιού είναι i:112 και j:287 τότε το πιόνι βρίσκεται στο κουτί ¨1¨.

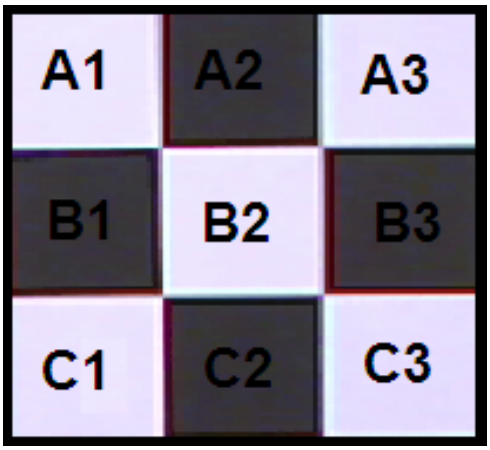


Εικόνα 3.6.2.2 Ονομασίες κουτιών σκακιέρας

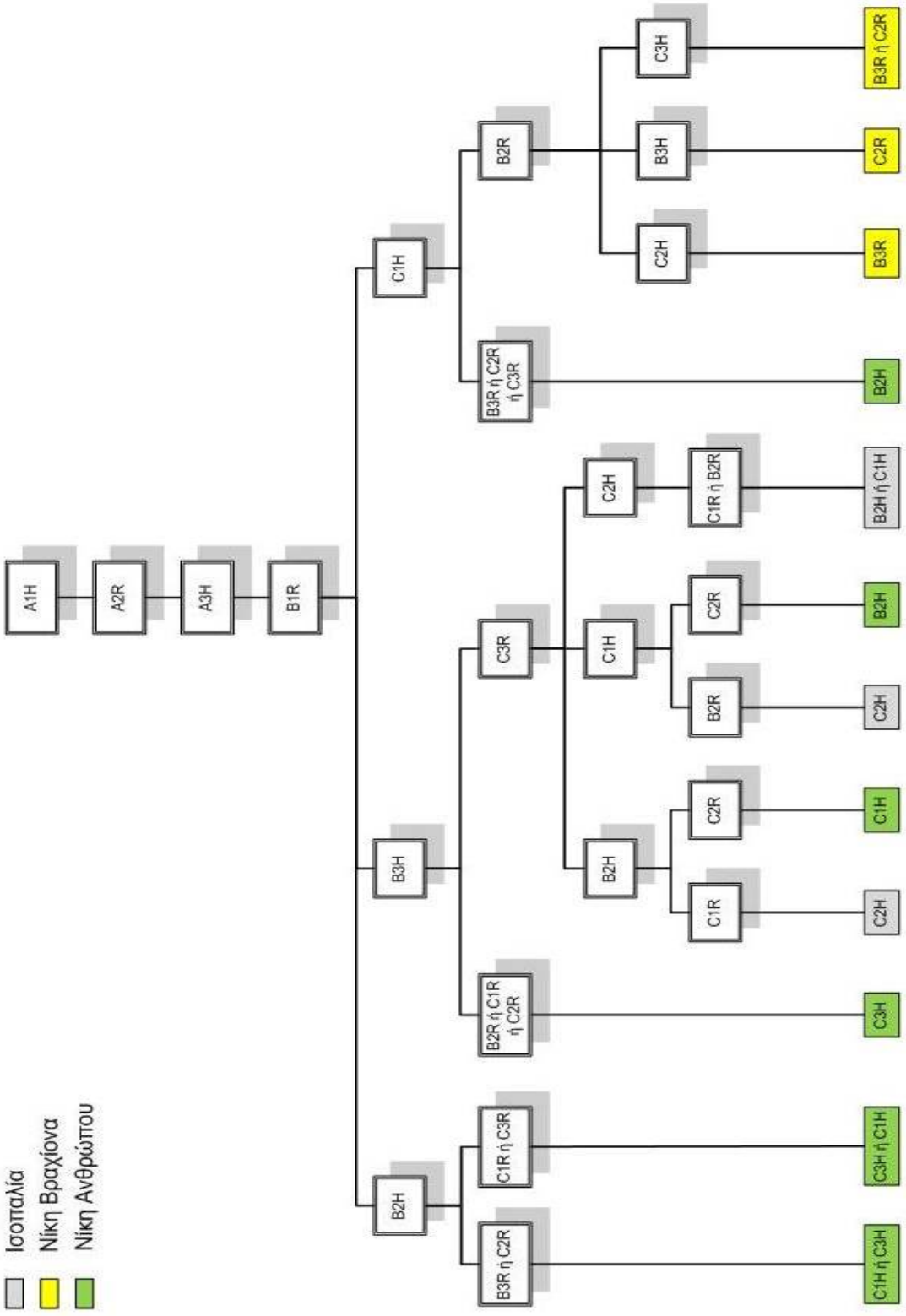
3.6.3 ΛΗΨΗ ΑΠΟΦΑΣΕΩΝ

Αφού είμαστε σε θέση να γνωρίζουμε αν έχει γίνει κάποια αλλαγή, σε ποιο κουτί της σκακιέρας έχει τοποθετηθεί το πιόνι αλλά και τις συντεταγμένες του πιονιού στον χώρο, τότε το σύστημα πρέπει σύμφωνα με μία σειρά κανόνων του παιχνιδιού, να επιλέξει σε ποιο κουτί της σκακιέρας θα πρέπει να τοποθετήσει ο βραχίονας το πιόνι του. Δίνουμε μία ονομασία σε κάθε ένα κουτί όπως φαίνεται στην παρακάτω εικόνα 3.6.3.1.

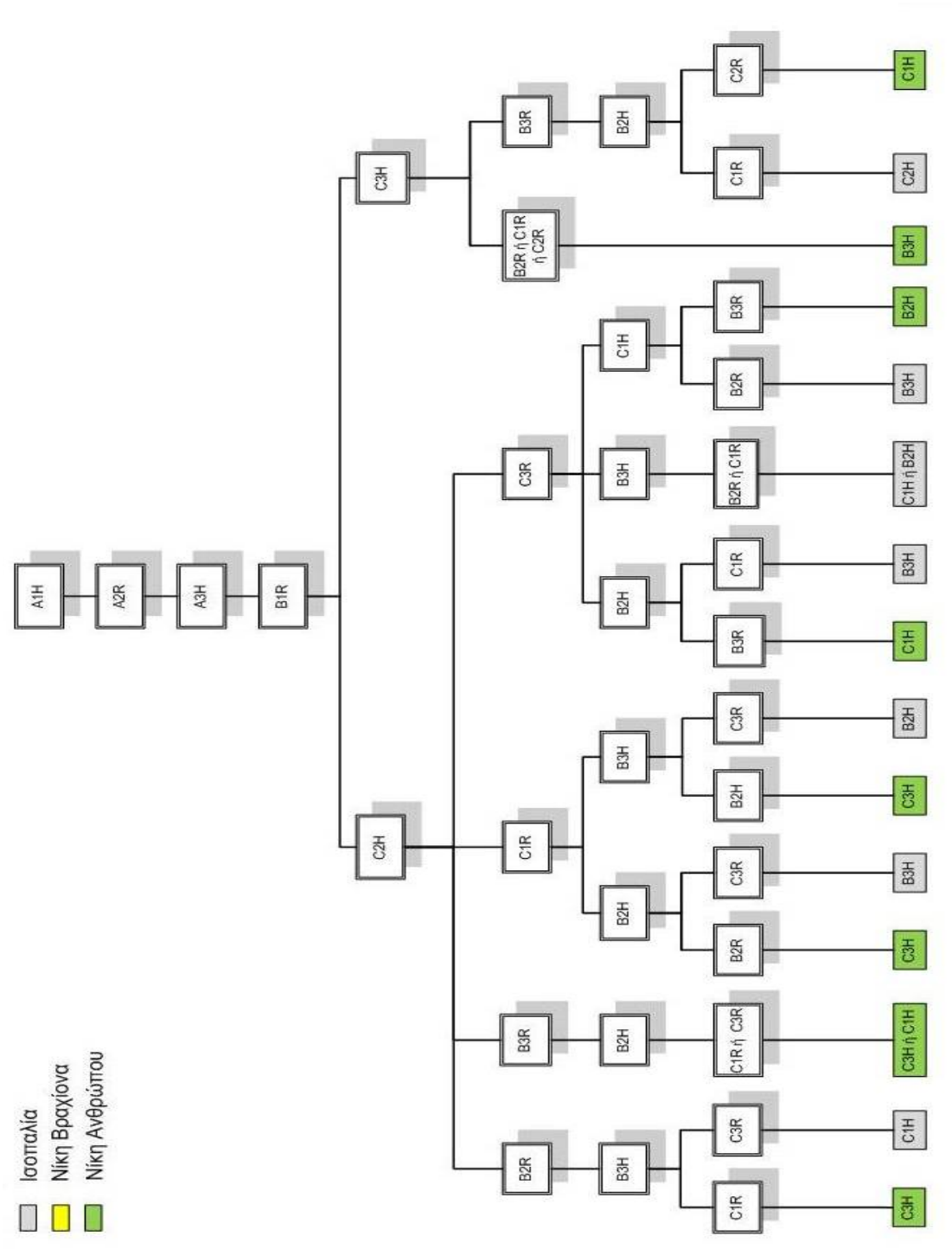
Οι κανόνες αυτοί έχουν την μορφή δεντοδιαγραμμάτων, το γράμμα ¨H¨ δείχνει ότι η κίνηση γίνεται από τον άνθρωπο και το γράμμα ¨R¨ ότι η κίνηση γίνεται από τον ρομποτικό βραχίονα. Για το παιχνίδι της τρίλιζας η λήψη των αποφάσεων αυτών γίνονται σύμφωνα με τα παρακάτω δεντρο-διαγράμματα.:



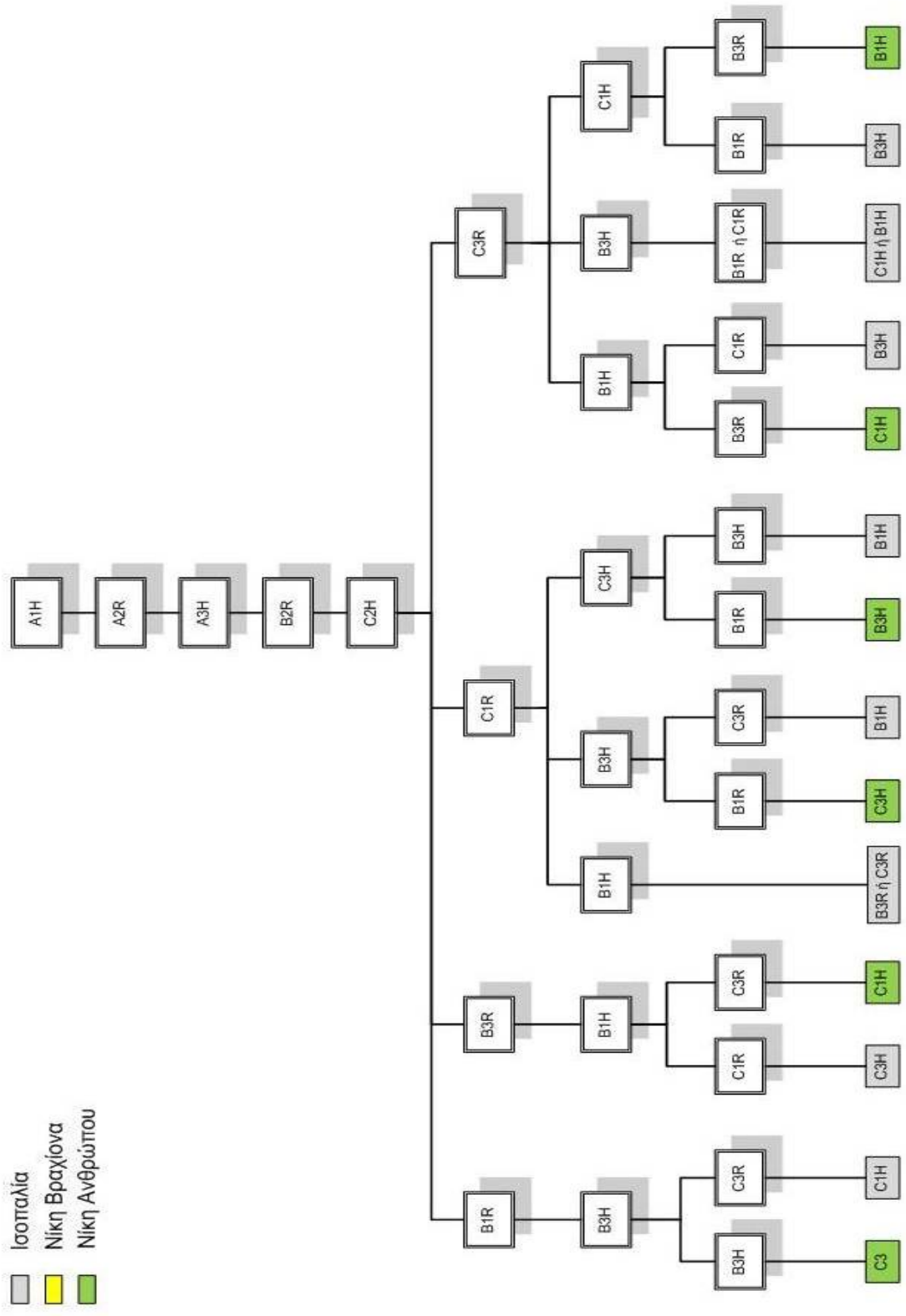
Εικόνα 3.6.3.1 Ονομασίες κουτιών σκακιέρας


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
B1R
C1H
B3H
B2H
B2R
B3R ή C2R ή C3R
C3H
B3H
C2H
B3R ή C2R
C2R
B3R
B2H
C2H
C1R ή B2R
B2H ή C1H
C3R
C1H
C2R
B2H
B2R
C2H
B2H
C2R
C1R
C1H
C2H
B2R ή C1R ή C2R
C3H
C1R ή C3R
C3H ή C1H
B3R ή C2R
C1H ή C3H

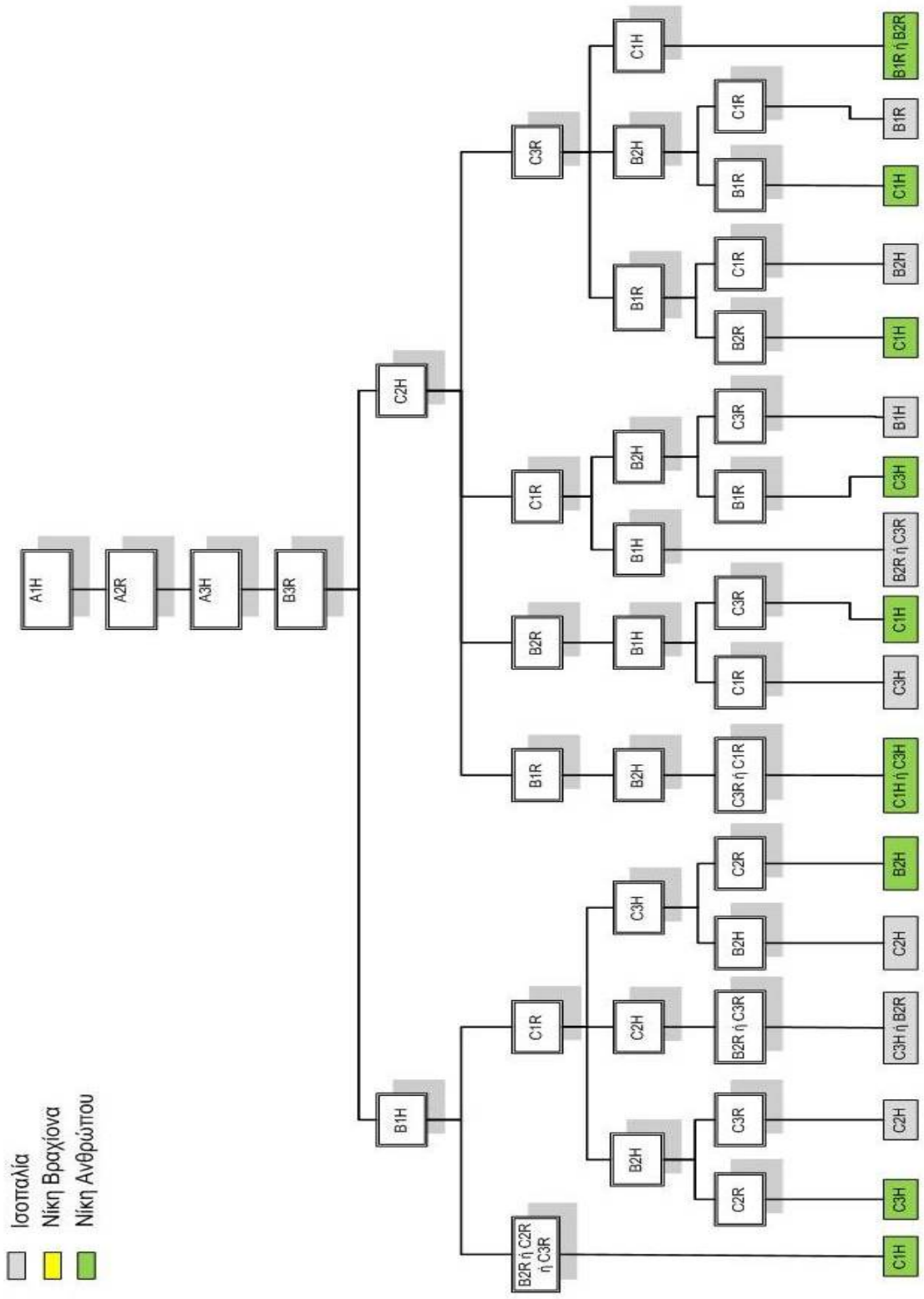
Σχήμα 3.6.3.1


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
B1R
C3H
C2H
B3R
B2R ή C1R ή C2R
C3R
C1R
B2R
B2H
B3H
C2R
C1R
C1H
B3R
B2R
B2R ή C1R
C1R
B3R
C3R
B2R
C3R
B2R
C1R ή C3R
C3R
C1R
C1H
C2H
B3H
B2H
B3H
B2H
B3H
C1H ή B2H
B3H
C1H
B2H
C3H
B3H
C3H
C3H ή C1H
C1H
C3H

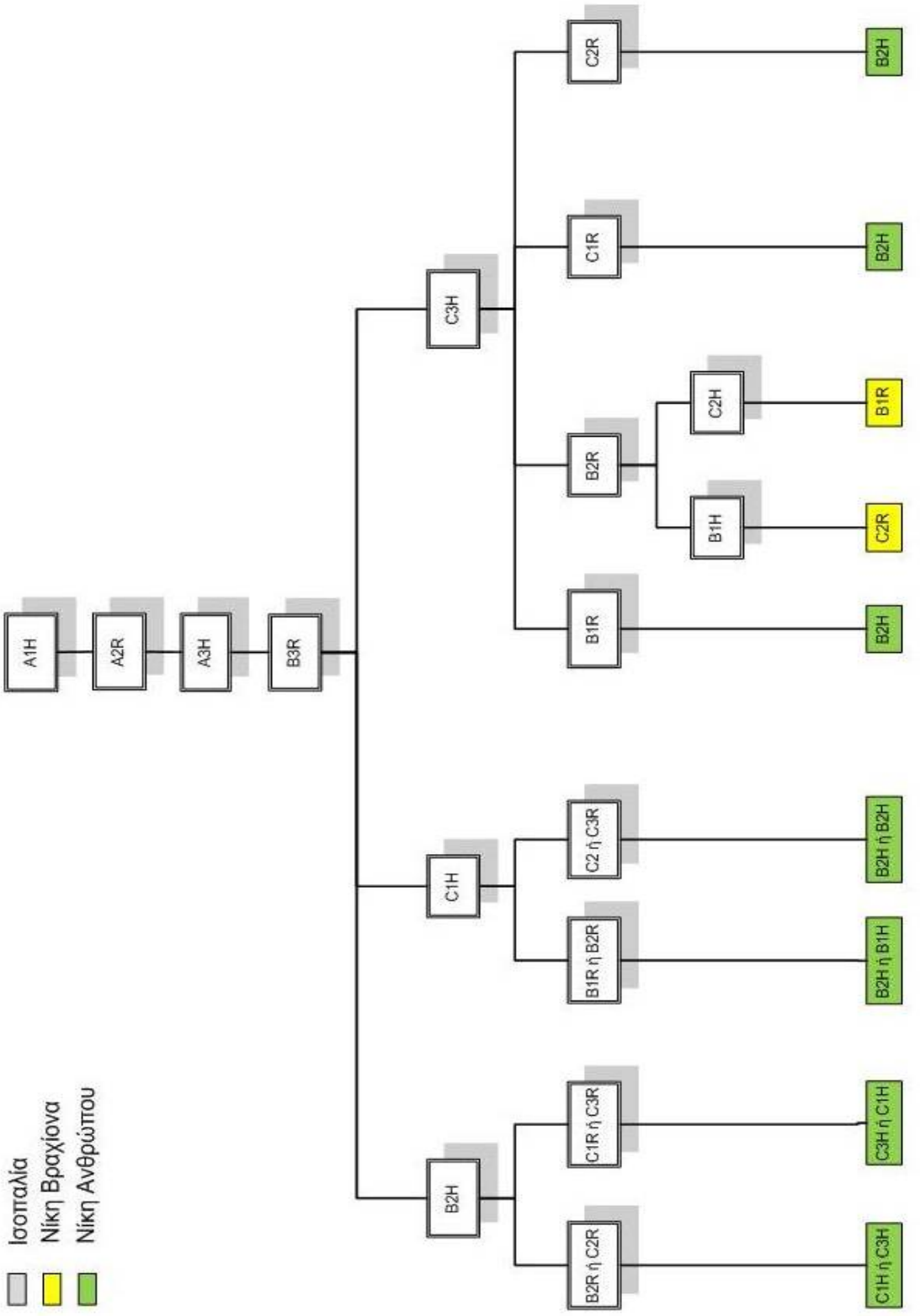
Σχήμα 3.6.3.2


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
B2R
C2H
C3R
C1R
B3R
B1R
C1H
B3H
B1H
C3H
B1R ή C1R
C1H ή B1H
B3R ή C3R
C3
B1H
B3H
C1H
C3H
C3R
C1R
B1R
B3H

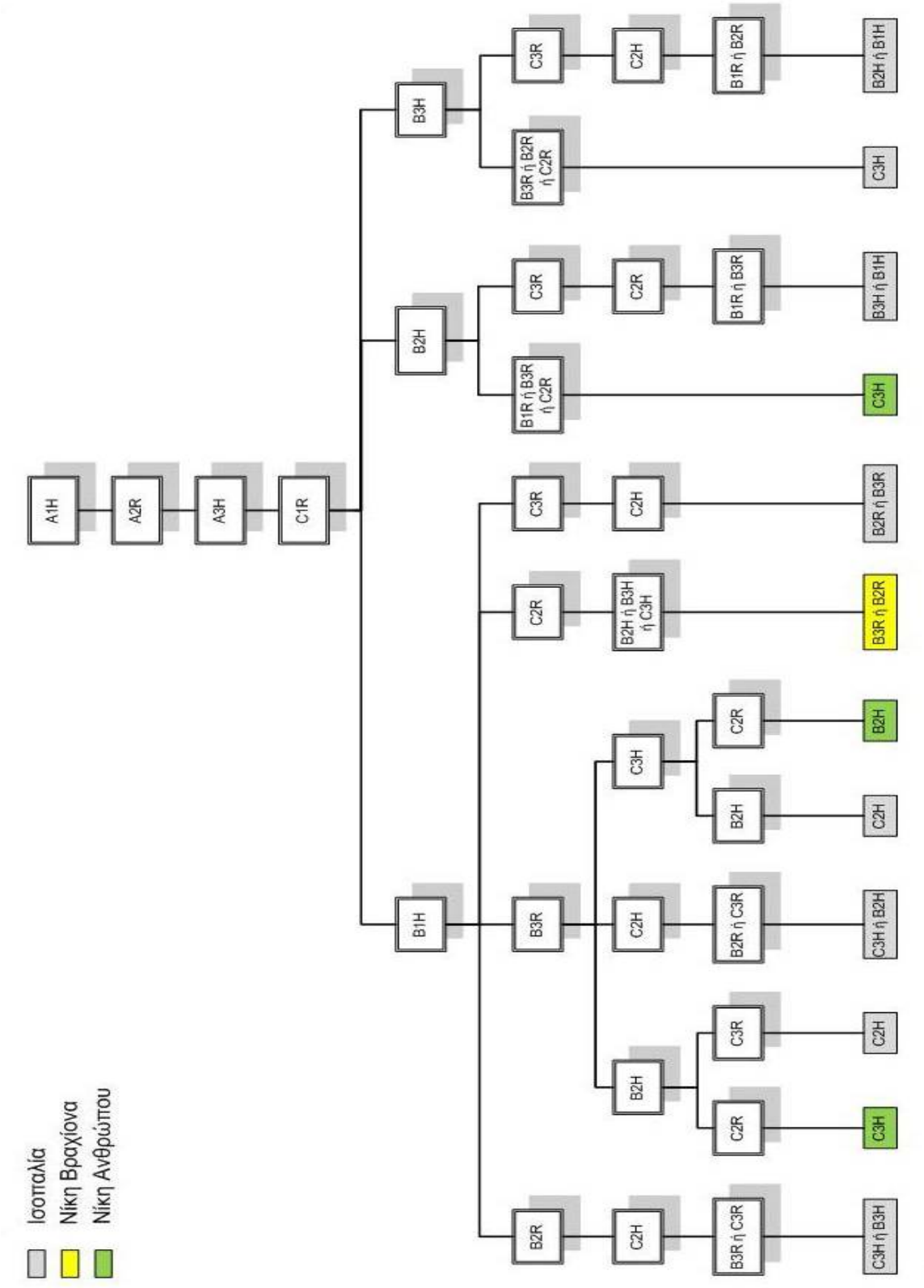
Σχήμα 3.6.3.3


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
B3R
C2H
B1H
C3R
C1R
B2R
B1R
B2H
B1H
C2R
C3H
B2H
C1R
C2H
B2H
C3R
C2R
B2R ή C2R ή C3R
C1H
B2H
B1R
C1R
B1R
B2H
C1H
B1R ή B2R
B1R
C1H
B2H
C1H
C3H
B1H
B2R ή C3R
C1H
C3H
C1H ή C3H
B2H
C2H
C3H ή B2R
C2H
C3H
C1H

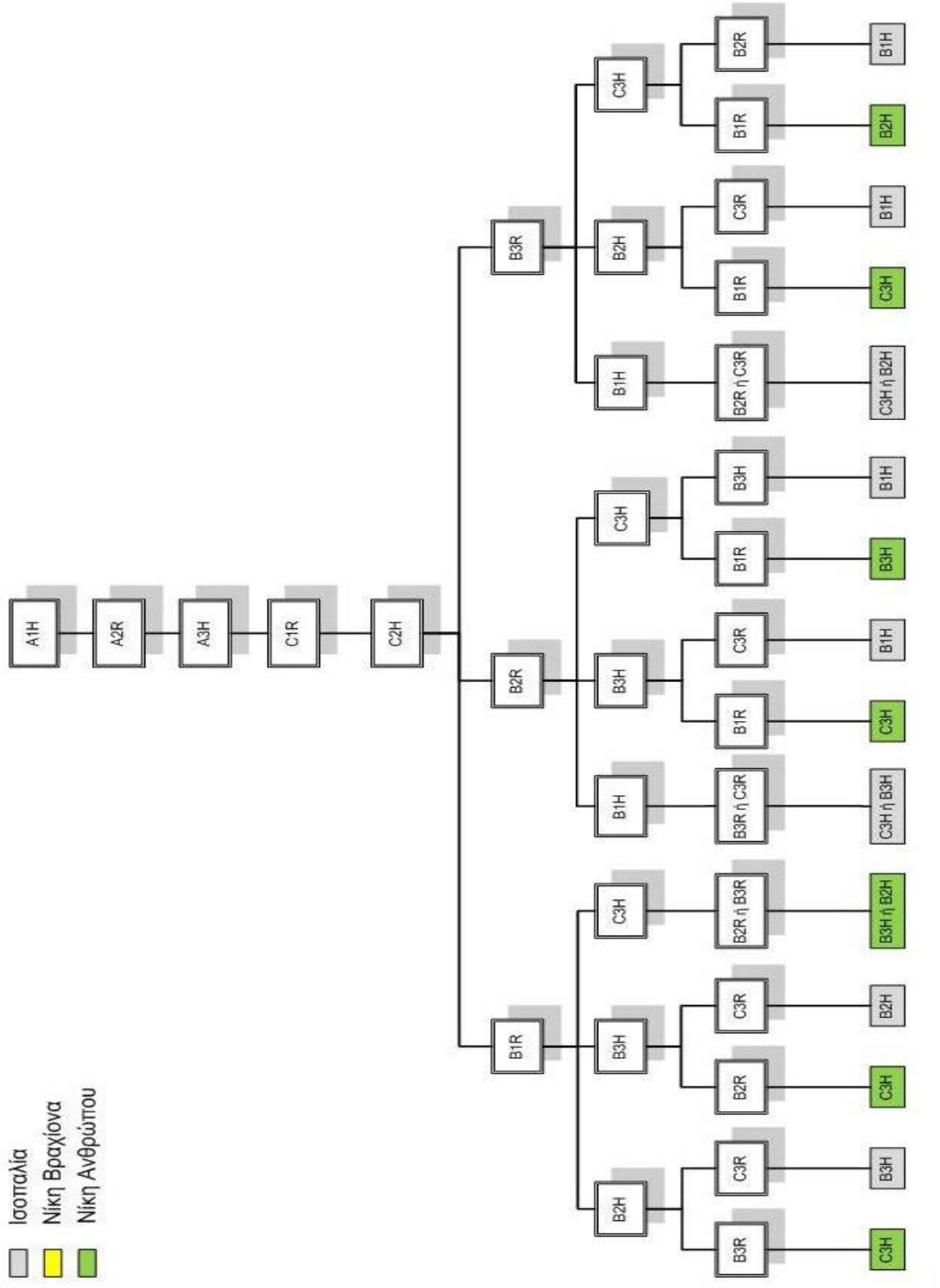
Σχήμα 3.6.3.4


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
B3R
C3H
C1H
B2H
C2R
C1R
B2R
B1R
C2H
B1H
C2 ή C3R
B1R ή B2R
C1R ή C3R
B2R ή C2R
B2H
B2H
B2H
B1R
C2R
B2H
B2H ή B2H
B2H ή B1H
C3H ή C1H
C1H ή C3H

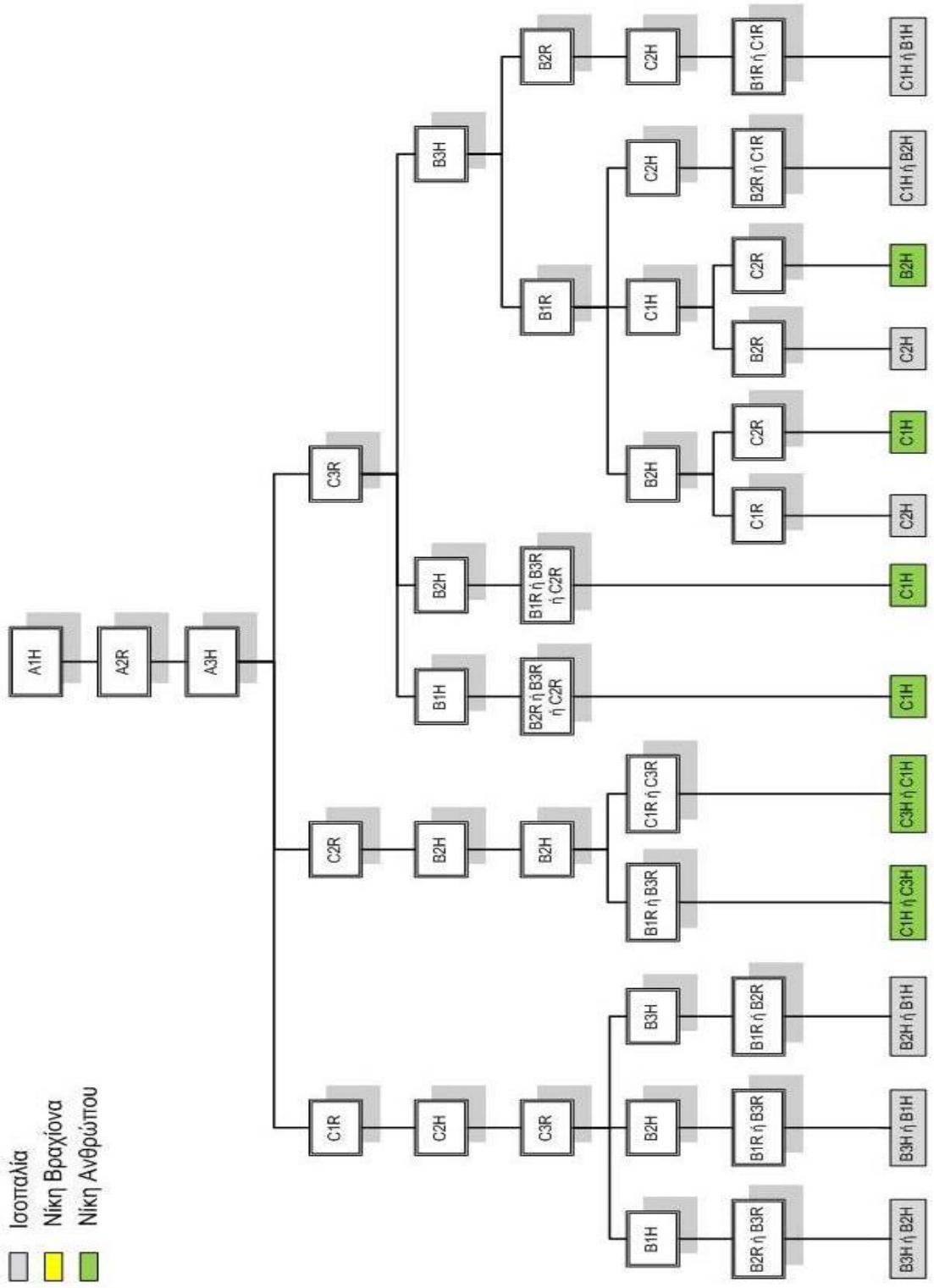
Σχήμα 3.6.3.5


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
C1R
B3H
B2H
B1H
C3R
B3R ή B2R ή C2R
B1R ή B3R ή C2R
C2R
B2R
C2H
B3R
B2H
C3H
B1R ή B2R
B2H ή B1H
C3H
B1R ή B3R
B3H ή B1H
C3H
B2R ή B3R
B3R ή B2R
B2H ή B3H ή C3H
B2H
C2H
B2R ή C3R
C3H ή B2H
C3R
C2H
C2R
C3H
B3R ή C3R
C3H ή B3H

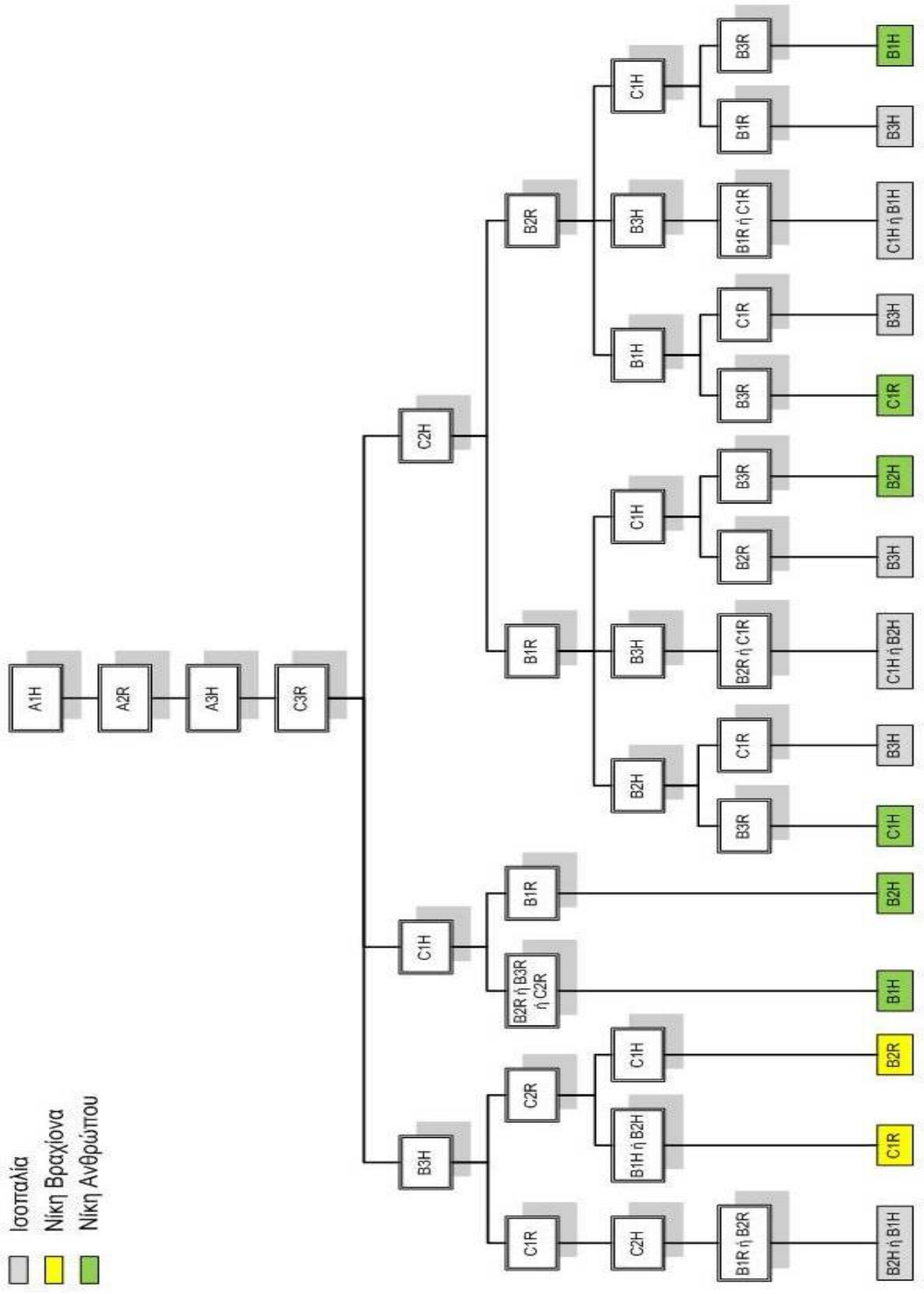
Σχήμα 3.6.3.6


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
C1R
C2H
B3R
B2R
B1R
C3H
B2H
B1H
B3H
B1R
C3R
B2R
B2R ή C3R
B3R ή C3R
B2R ή B3R
B1H
B2H
C3H
C3H ή B2H
B3H
C3H ή B3H
B3H ή B2H

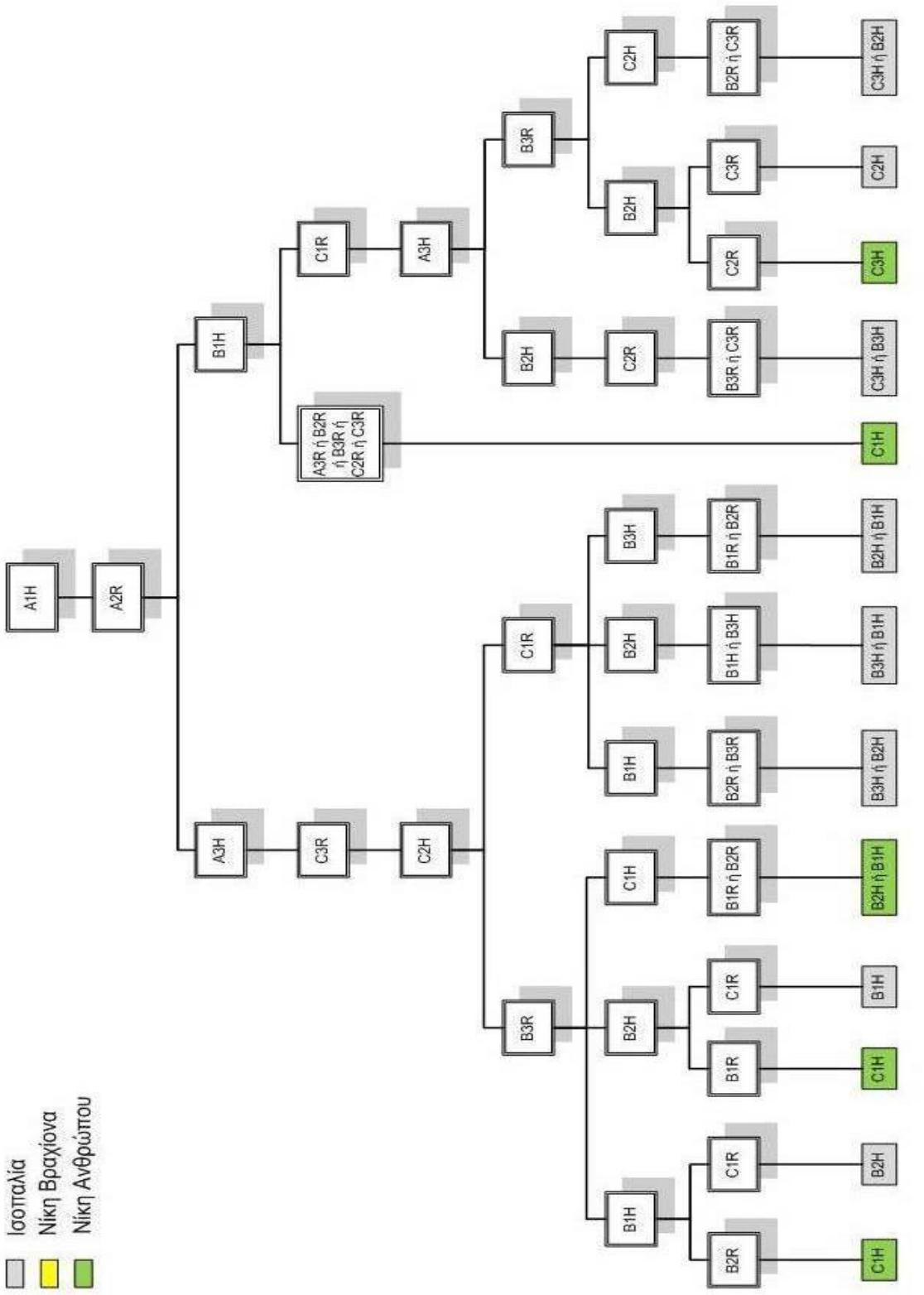
Σχήμα 3.6.3.7


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
C3R
B3H
B2R
C2H
B1R ή C1R
C1H ή B1H
C2H
B2R ή C1R
C1H ή B2H
B1R
C1H
C2R
B2H
C2H
B2R
C2H
B2H
C2R
C1H
C1R
C2H
B2H
B1R ή B3R ή C2R
C1H
B1H
B2R ή B3R ή C2R
C1H
C2R
B2H
B2H
C1R ή C3R
C3H ή C1H
B1R ή B3R
C1H ή C3H
C1R
C2H
C3R
B3H
B1R ή B2R
B2H ή B1H
B2H
B1R ή B3R
B3H ή B1H
B1H
B2R ή B3R
B3H ή B2H

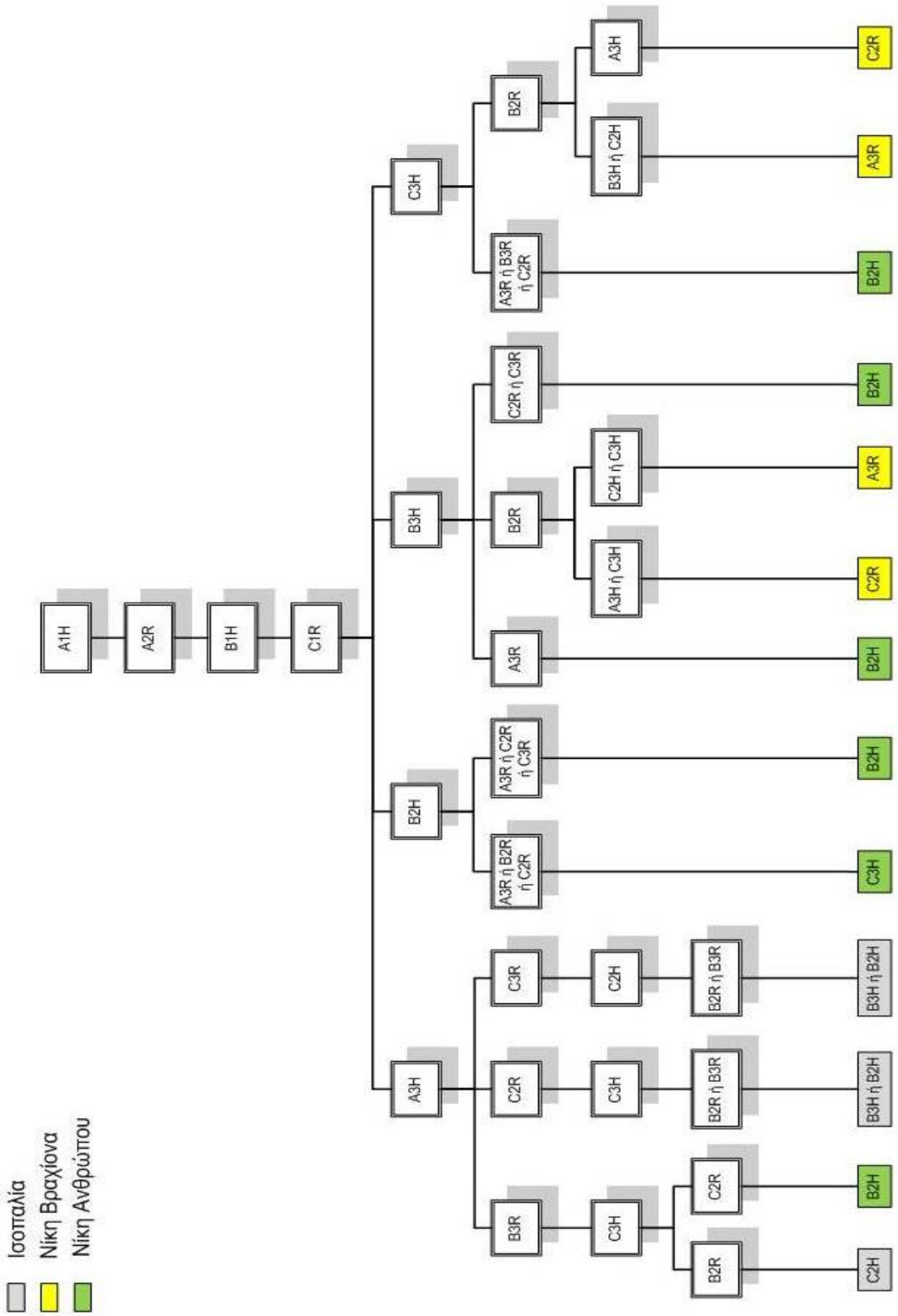
Σχήμα 3.6.3.8


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
A3H
C3R
C2H
B2R
C1H
B3R
B1H
B3H
B1R ή C1R
C1H ή B1H
B1R
B3H
C1R
B3H
B3R
C1R
B2H
B3R
B2R
B3H
B1R
B3H
B2R ή C1R
C1H ή B2H
B2H
C1R
B3H
B3R
C1H
C1H
B1R
B2H
B2R ή B3R ή C2R
B1H
B3H
C2R
C1H
B2R
B1H ή B2H
C1R
C1R
C2H
B1R ή B2R
B2H ή B1H

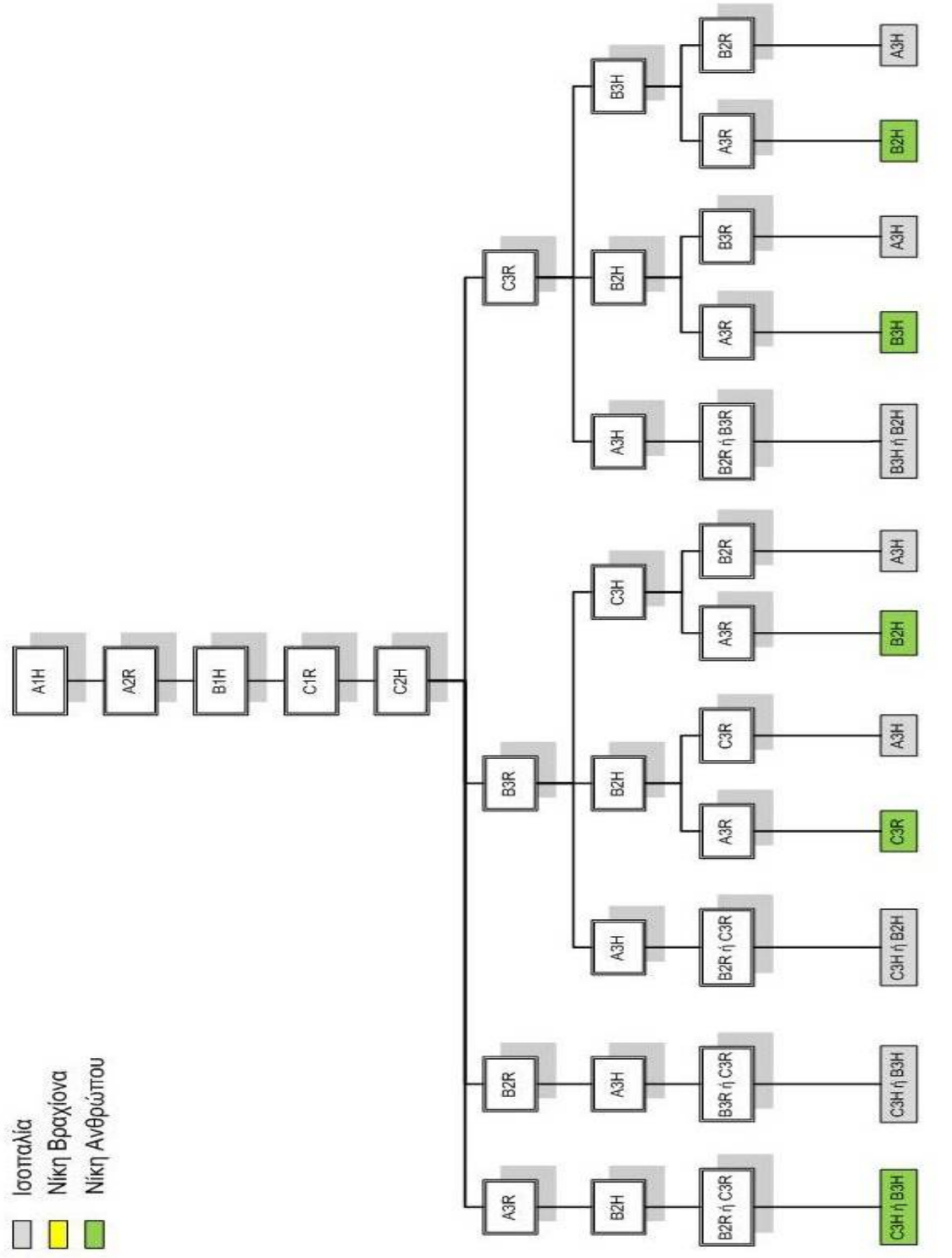
Σχήμα 3.6.3.9


Ισοταλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B1H
A3H
C1R
A3H
A3R ή B2R ή ή B3R ή C2R ή C3R
C3R
C2H
B3R
B2H
C2H
B2H
C2R
C3R
C2R
B2R ή C3R
B3R ή C3R
C3H ή B2H
C2H
C3H
C3H ή B3H
C1H
C1R
B3H
B2H
B1H
B1R ή B2R
B1H ή B3H
B2R ή B3R
B2H ή B1H
B3H ή B1H
B3H ή B2H
B3R
C1H
B2H
B1H
B1R ή B2R
B2H ή B1H
C1R
B1R
C1H
B1H
C1R
B2R
B2H
C1H

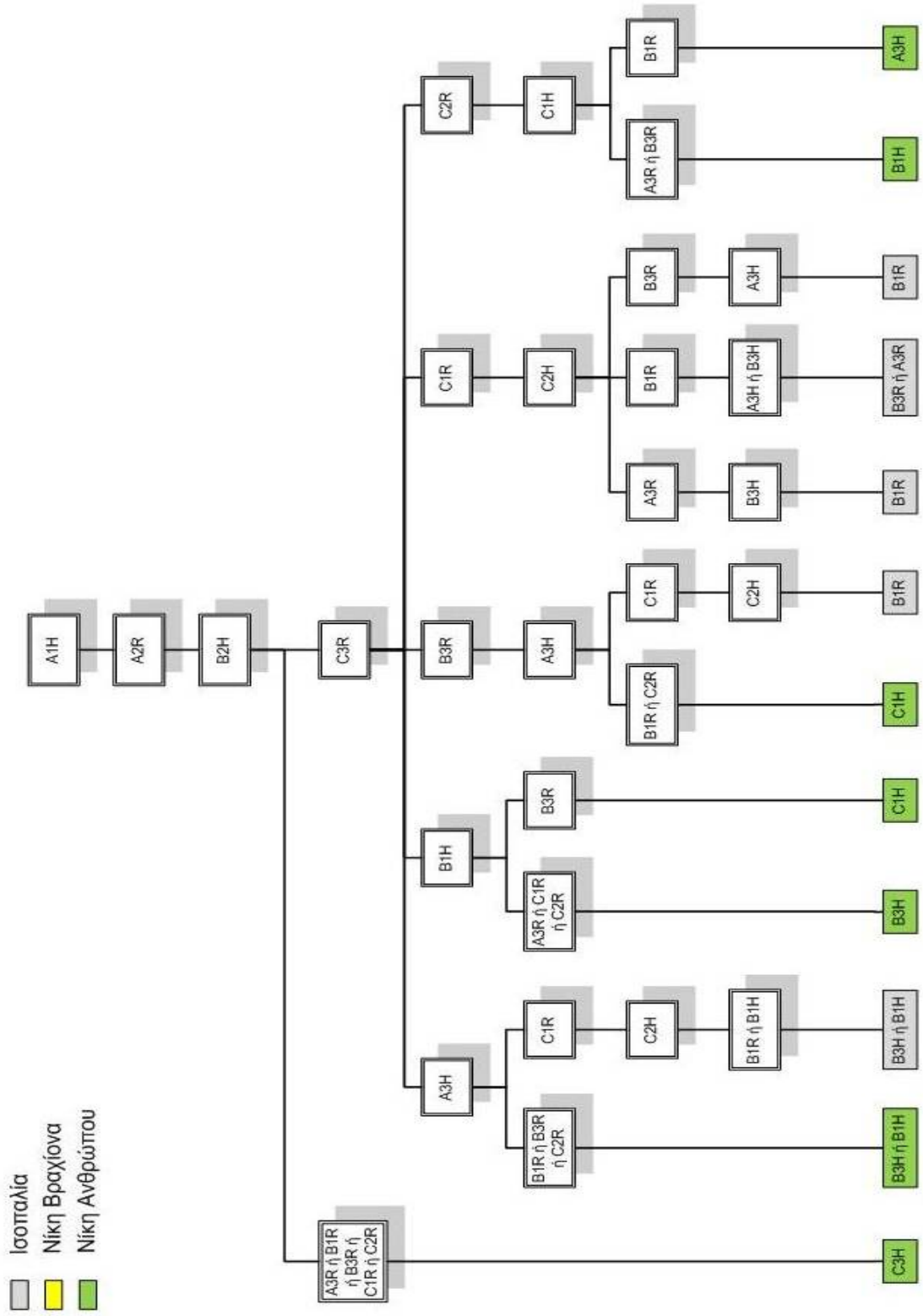
Σχήμα 3.6.3.10


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B1H
C1R
A3H
B3R
C3H
C2R
B2R
C3R
C2H
B2H
A3R ή B2R ή C2R
A3R ή C2R ή C3R
B2H
A3H
B2R
C3H
A3R ή B3R ή C2R
B3H ή C2H
A3H
C2R
A3R
B2H
A3R
A3H ή C3H
C2H ή C3H
B2R
C2R ή C3R
B2H
C2R
B2H
C3H
B2H
B2R ή B3R
B3H ή B2H
B2R ή B3R
B3H ή B2H
C2R
B2H
B2R
C2H

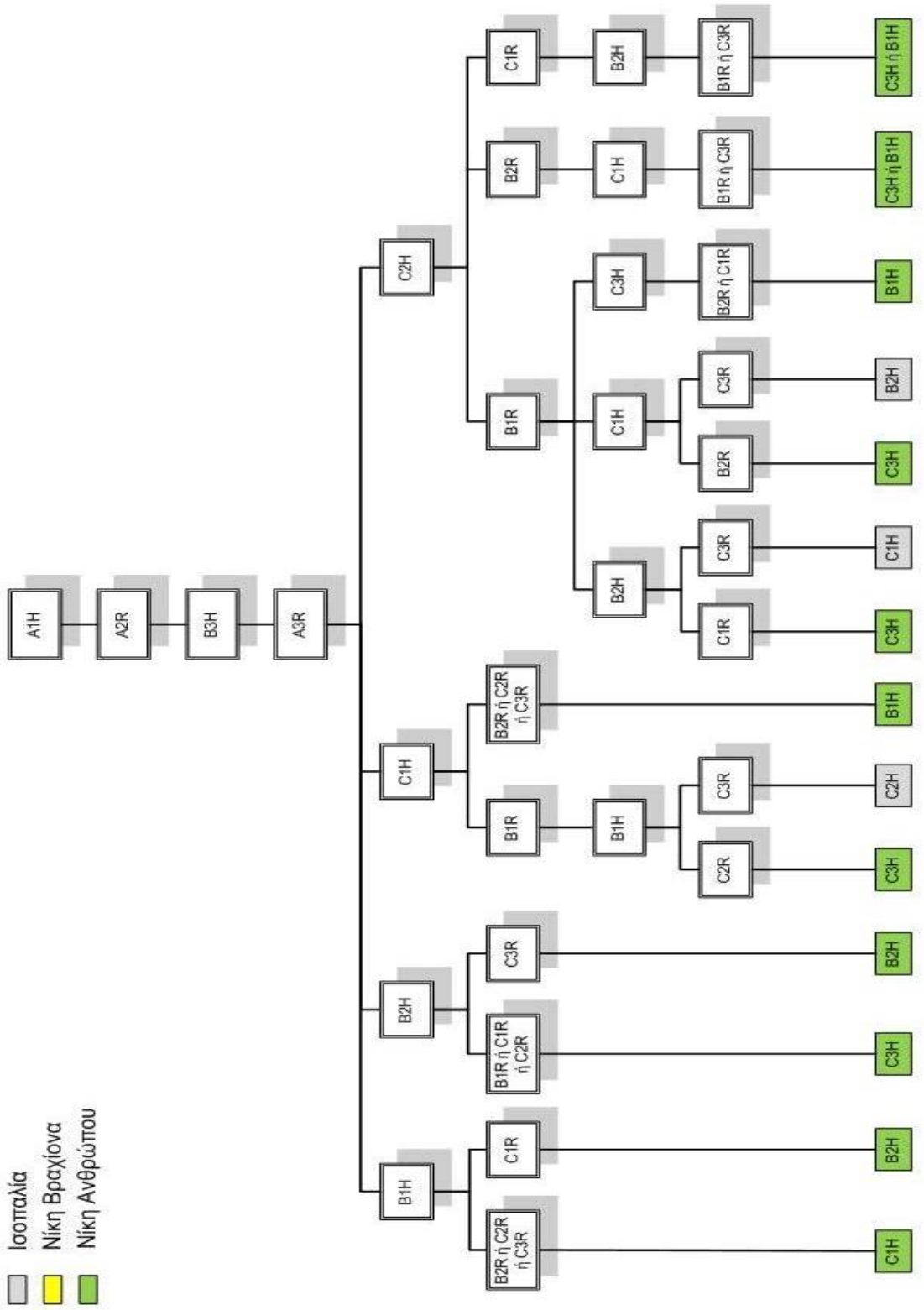
Σχήμα 3.6.3.11


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B1H
C1R
C2H
A3R
B2H
B2R ή C3R
C3H ή B3H
B2R
A3H
B3R ή C3R
C3H ή B3H
B3R
B2H
A3H
B2R ή C3R
C3H ή B2H
C3R
A3R
A3H
C3R
B2H
A3R
B2R
A3H
B2H
C3H
A3H
B2R ή B3R
B3H ή B2H
B3H
A3R
B3R
A3H
B3H
B2H
A3R
B2R
A3H
B2H

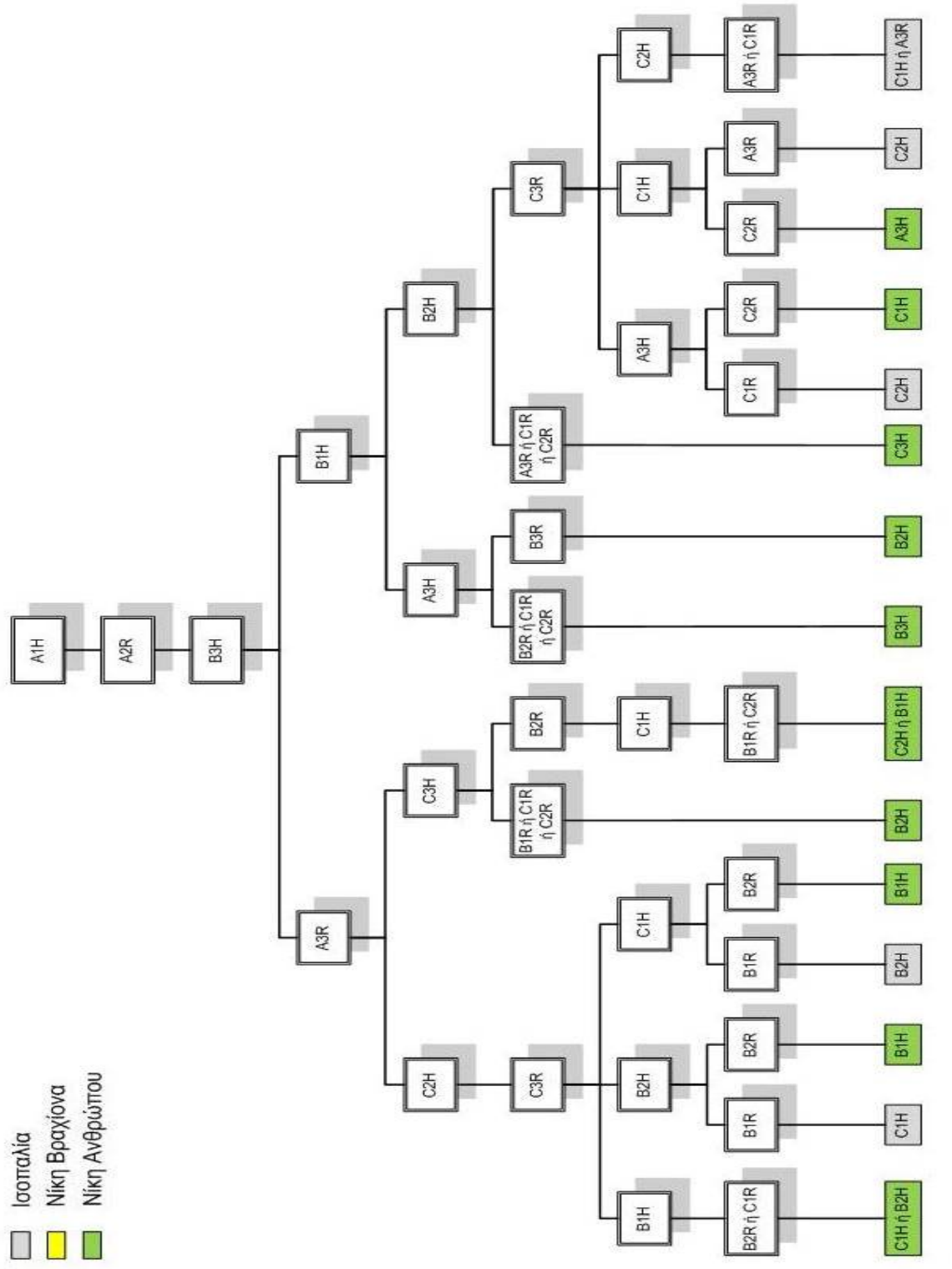
Σχήμα 3.6.3.12


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B2H
C3R
A3R ή B1R ή B3R ή C1R ή C2R
C3H
B3R
A3H
C1R
B1R ή C2R
C2H
B1R
C1H
B1R
A3R ή B3R
A3H
B1H
C1R
A3R
B3H
B1R
C2H
A3H ή B3H
B3R ή A3R
B1R
B3R
B1R
A3H
B1R
C1H
B1H
B3R
A3R ή C1R ή C2R
C1H
B3H
A3H
C1R
B1R ή B3R ή C2R
C2H
B1R ή B1H
B3H ή B1H
B3H ή B1H

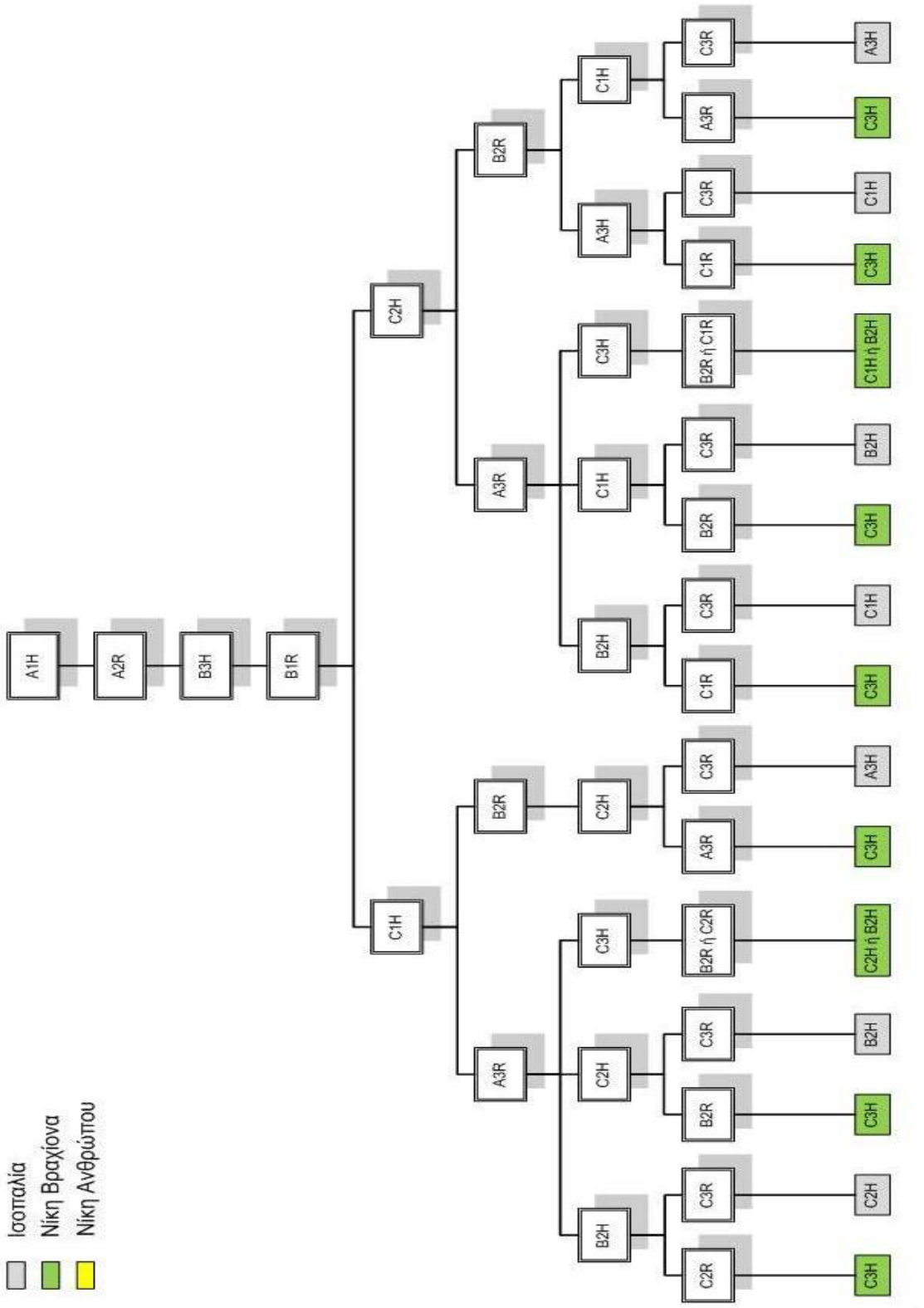
Σχήμα 3.6.3.13


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
A3R
C2H
C1R
B2H
B1R ή C3R
C3H ή B1H
B2R
C1H
B1R ή C3R
C3H ή B1H
B1R
C3H
B2R ή C1R
B1H
C1H
C3R
B2H
B2R
C3H
B2H
C3R
C1H
C1R
C3H
C1H
C3R
B2R ή C2R ή C3R
B1H
B1R
B1H
C3R
C2H
C2R
C3H
B2H
C3R
B1R ή C1R ή C2R
C3H
B1H
C1R
B2H
B2R ή C2R ή C3R
C1H

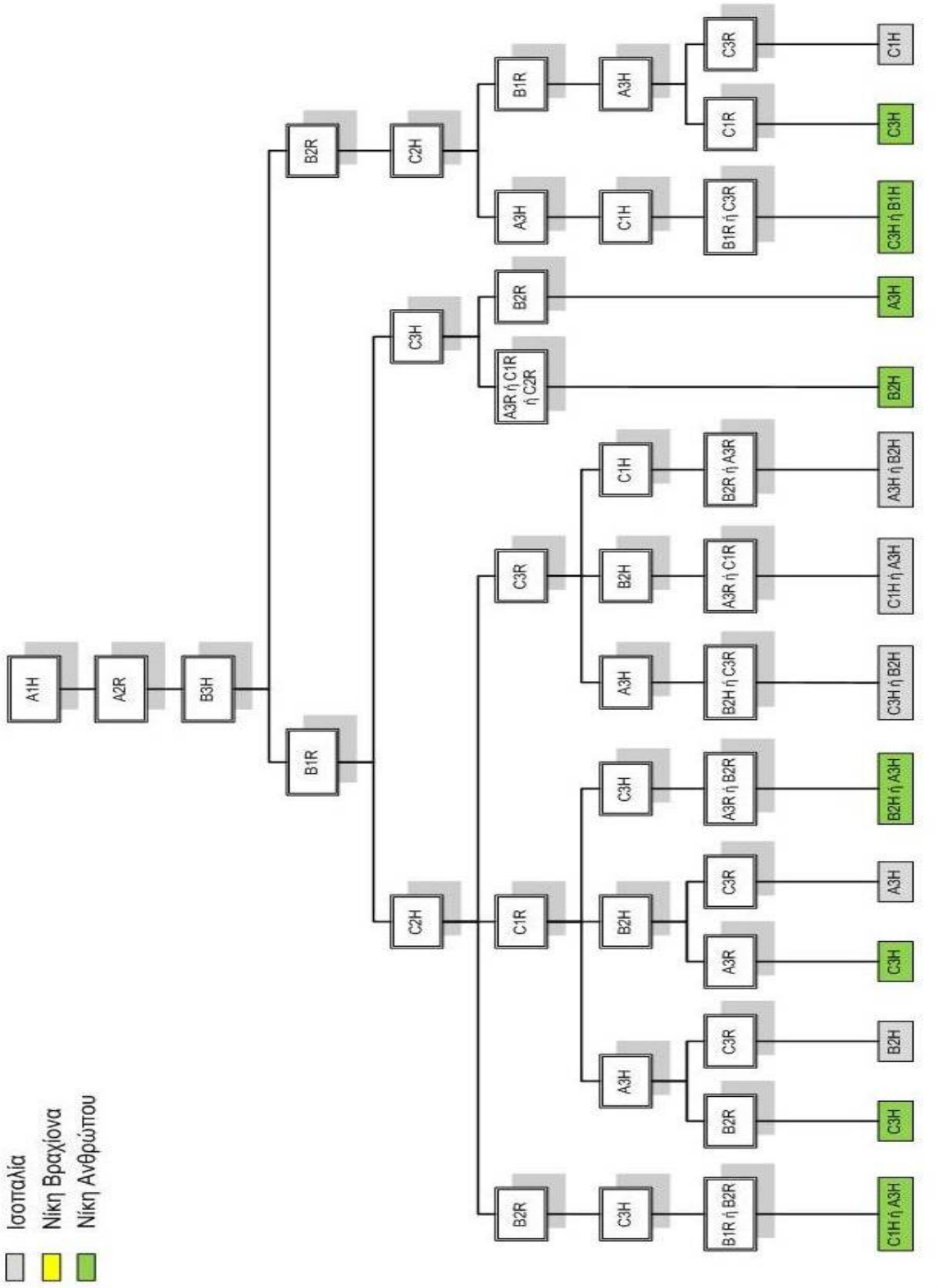
Σχήμα 3.6.3.14


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
B1H
A3R
B2H
A3H
C3H
C2H
B3R
B2R
A3R ή C1R ή C2R
B2R ή C1R ή C2R
B1R ή C1R ή C2R
C3R
C1H
A3H
C2R
C1R
A3R
B1R
B2H
B1H
B2R ή C1R
C1H ή A3R
C2H
A3H
C1H
C2H
C3H
B2H
B3H
C2H ή B1H
B2H
B1H
B2H
C1H
C1H ή B2H

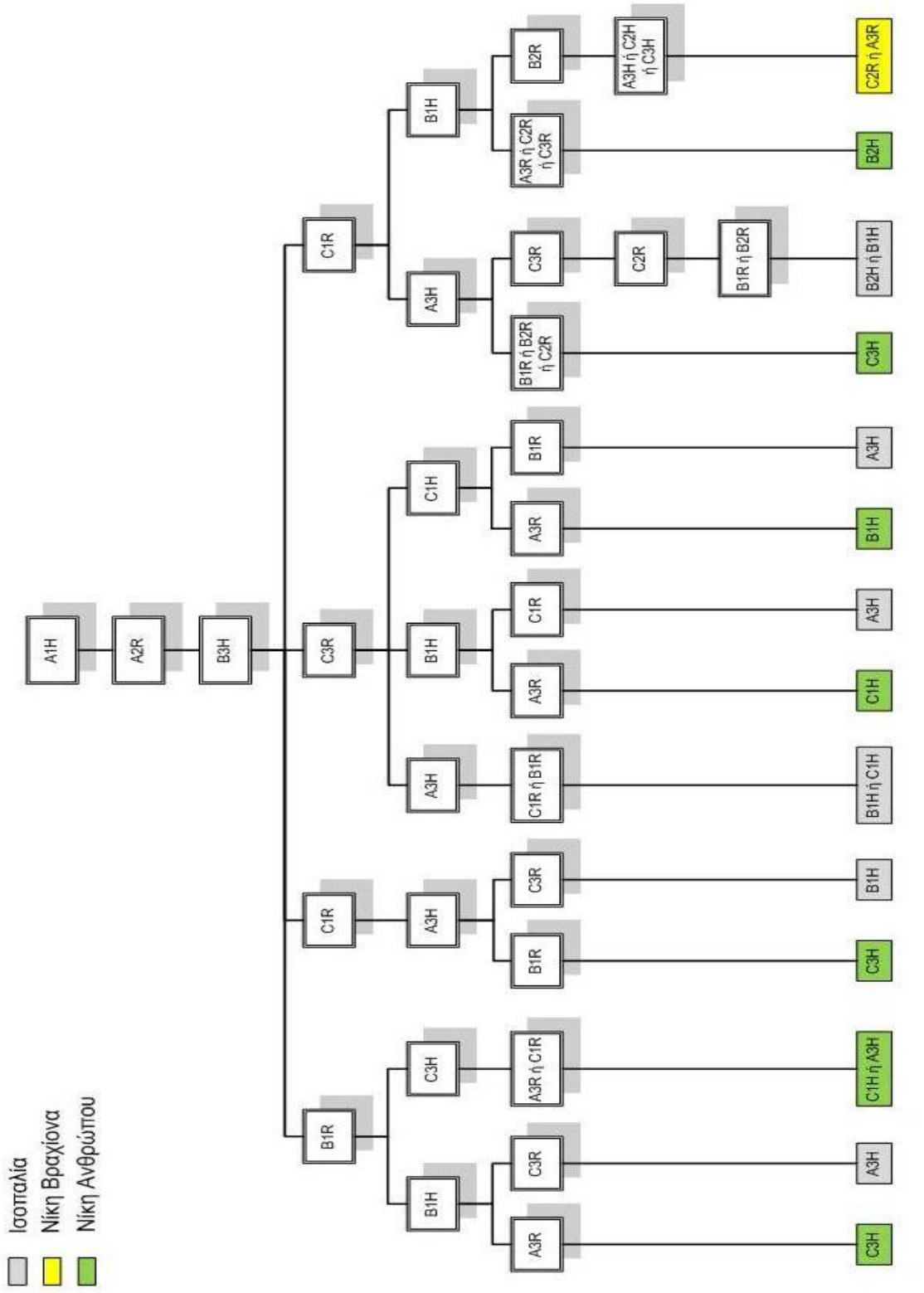
Σχήμα 3.6.3.15


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
B1R
C2H
C1H
B2R
A3R
C1H
A3H
C3H
B2R ή C1R
C2H
B2R ή C2R
C3R
C1R
B2R
A3R
C2R
B2H
C1H
C2H
C3H
A3H
B2H
C1H ή B2H
C2H ή B2H

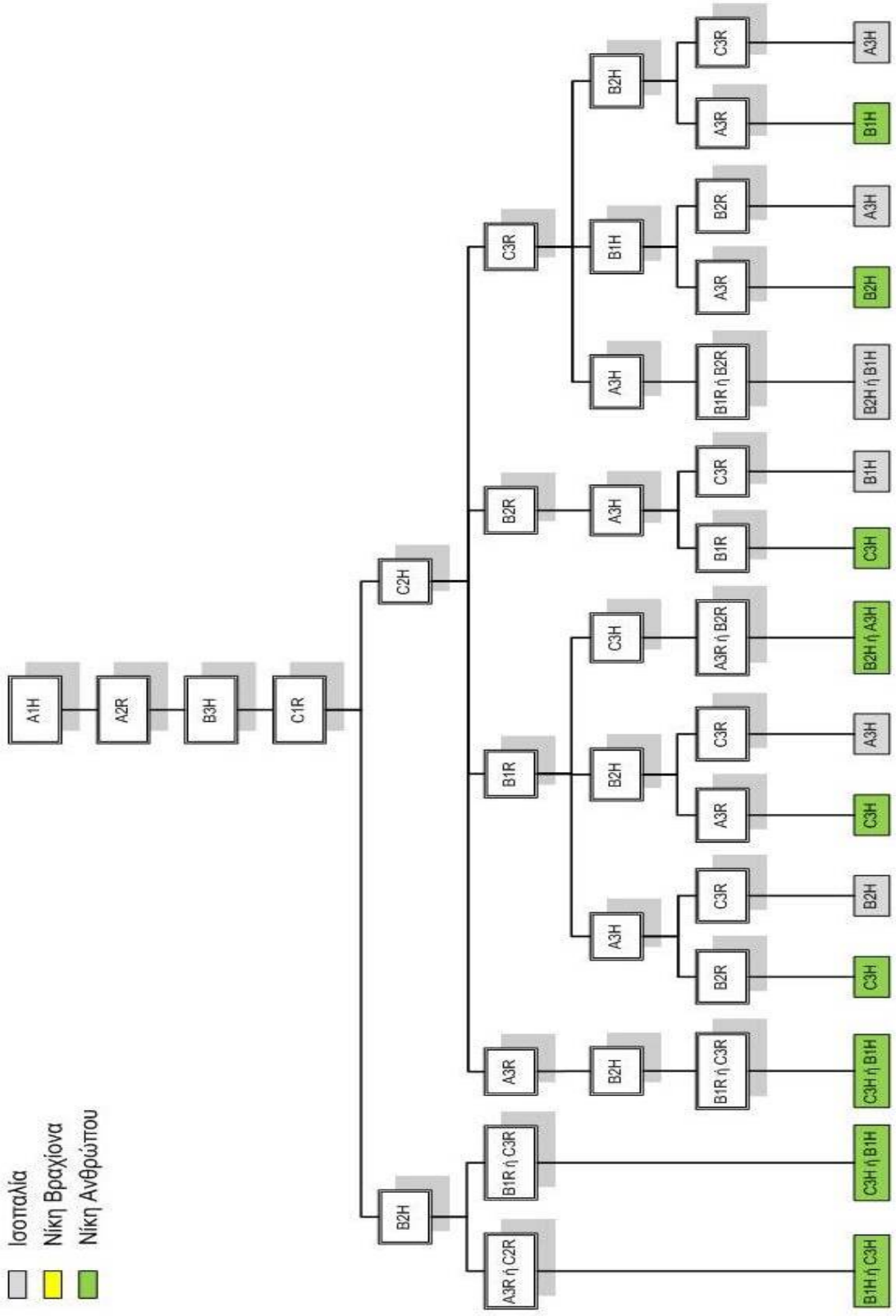
Σχήμα 3.6.3.16


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
B2R
B1R
C2H
C3H
C1R
C3R
B1R
A3H
A3H
C1H
B2R
A3R ή C1R ή C2R
C1H
B2H
A3H
C3R
C1R
B2H
A3R
A3H
C3H
B2R
C3H
C3R
B2R
C1H
C3H
B2H
C3H
A3H
B2H
C3H
A3H
B1R ή C3R
B2R ή A3R
A3R ή C1R
B2H ή C3R
A3R ή B2R
C3R
A3R
B2R
B1R ή B2R
C3H ή B1H
A3H
B2H
A3H ή B2H
C1H ή A3H
C3H ή B2H
B2H ή A3H
A3H
C3H
B2H
C3H
C1H ή A3H

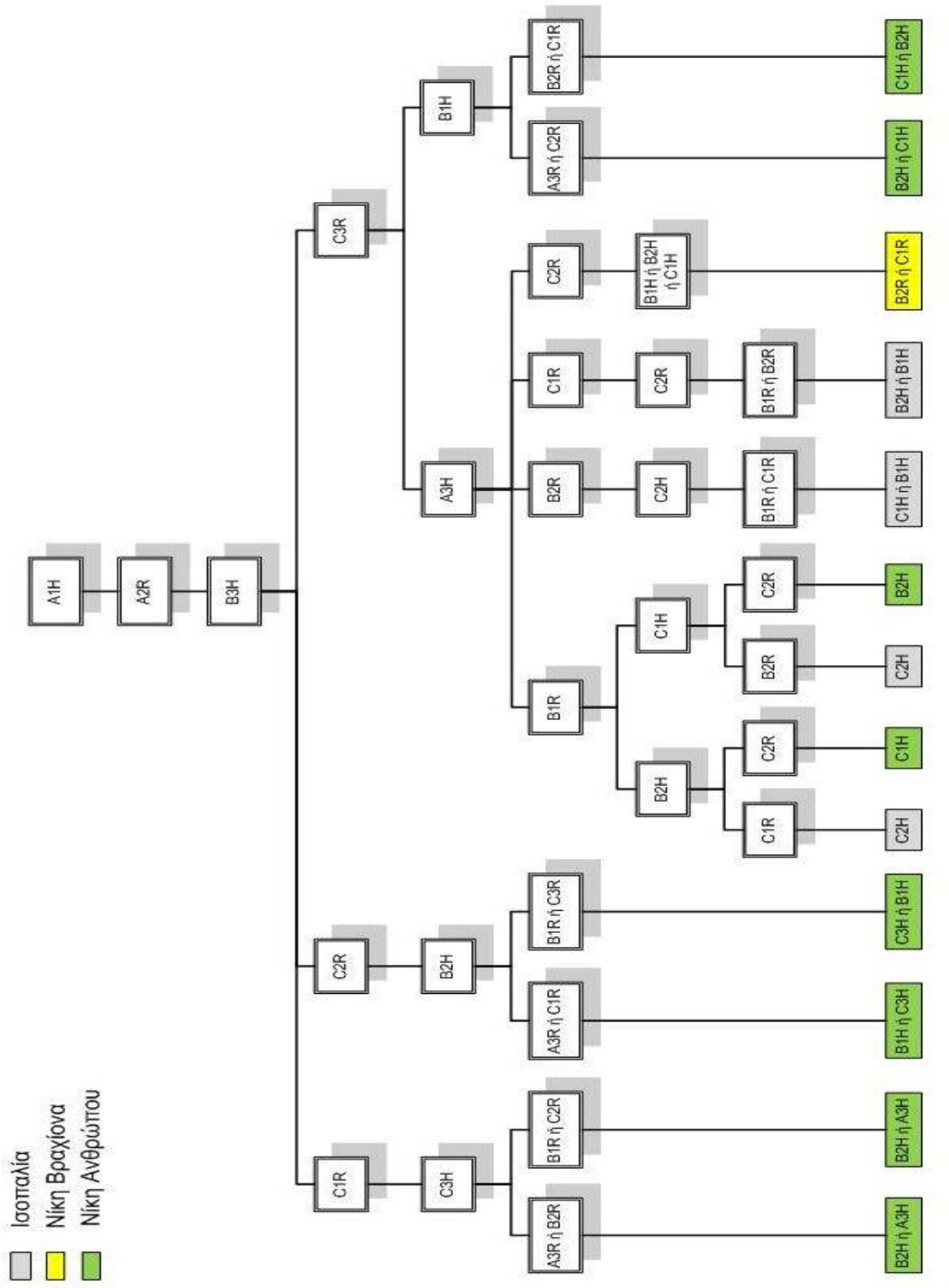
Σχήμα 3.6.3.17


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
C3R
C1R
B1R
B1H
A3H
C1H
A3R
C3H
B2R
A3H ή C2H ή C3H
C2R ή A3R
A3R ή C2R ή C3R
B2H
C3R
C2R
B1R ή B2R
B2H ή B1H
B1R ή B2R ή C2R
C3H
B1R
A3H
C1R
A3H
C1R ή B1R
B1H ή C1H
A3R ή C1R
C1H ή A3H

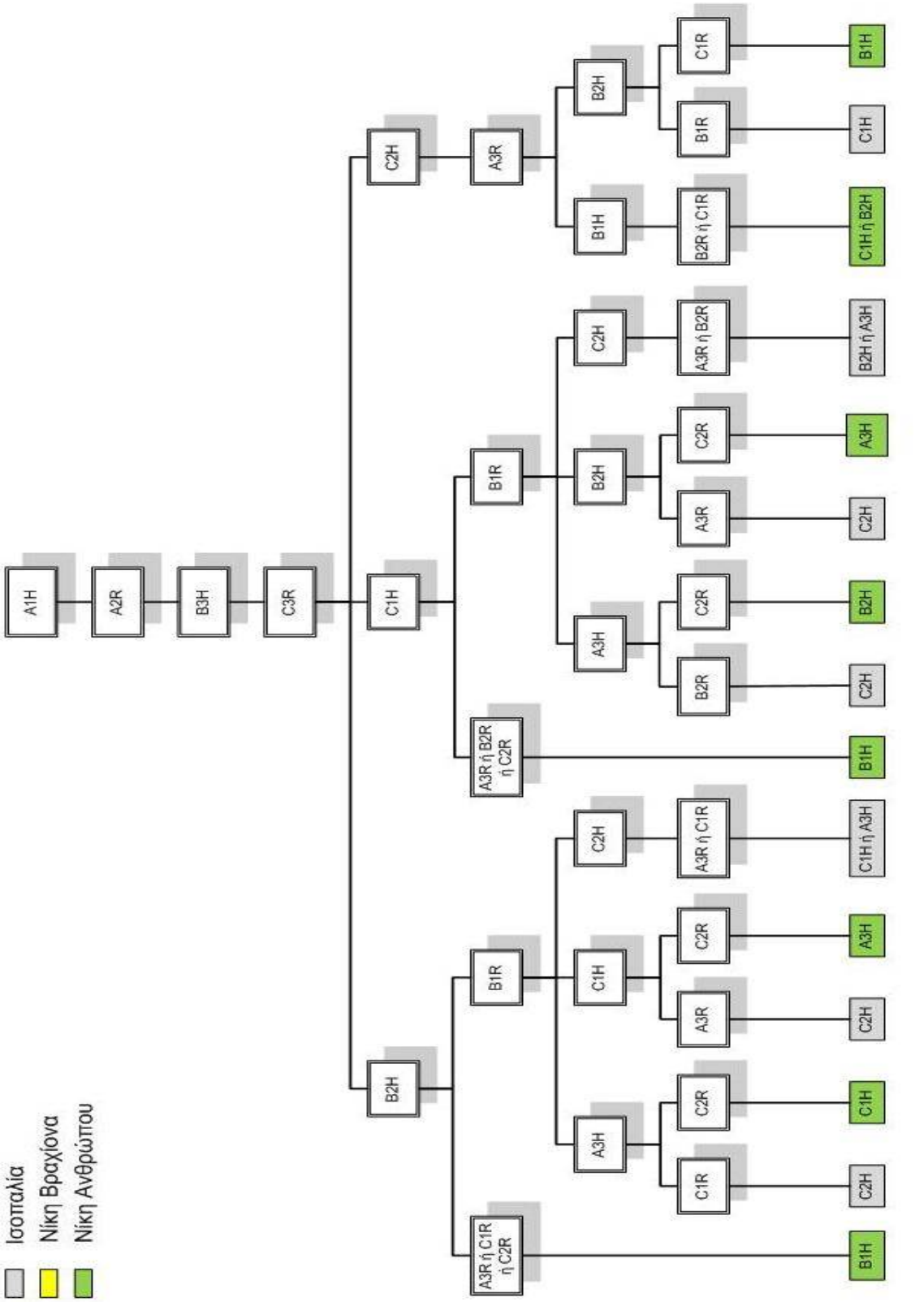
Σχήμα 3.6.3.18


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
C1R
C2H
B2H
C3R
B2R
B1R
A3R
B1R ή C3R
A3R ή C2R
B2H
A3H
B1H
B1R ή B2R
C3R
B2R
A3R
C3H
B1R
A3R ή B2R
C3H
B2H
A3R
A3H
B2R
C3H
B1R ή C3R
C3H ή B1H
C3H ή B1H
B1H ή C3H
A3H
B1H
B2H
A3H
B1H
B2H ή B1H
A3H
B2H
A3H
C3H
B2H ή A3H
A3H
C3H
B2H
C3H
C3H ή B1H

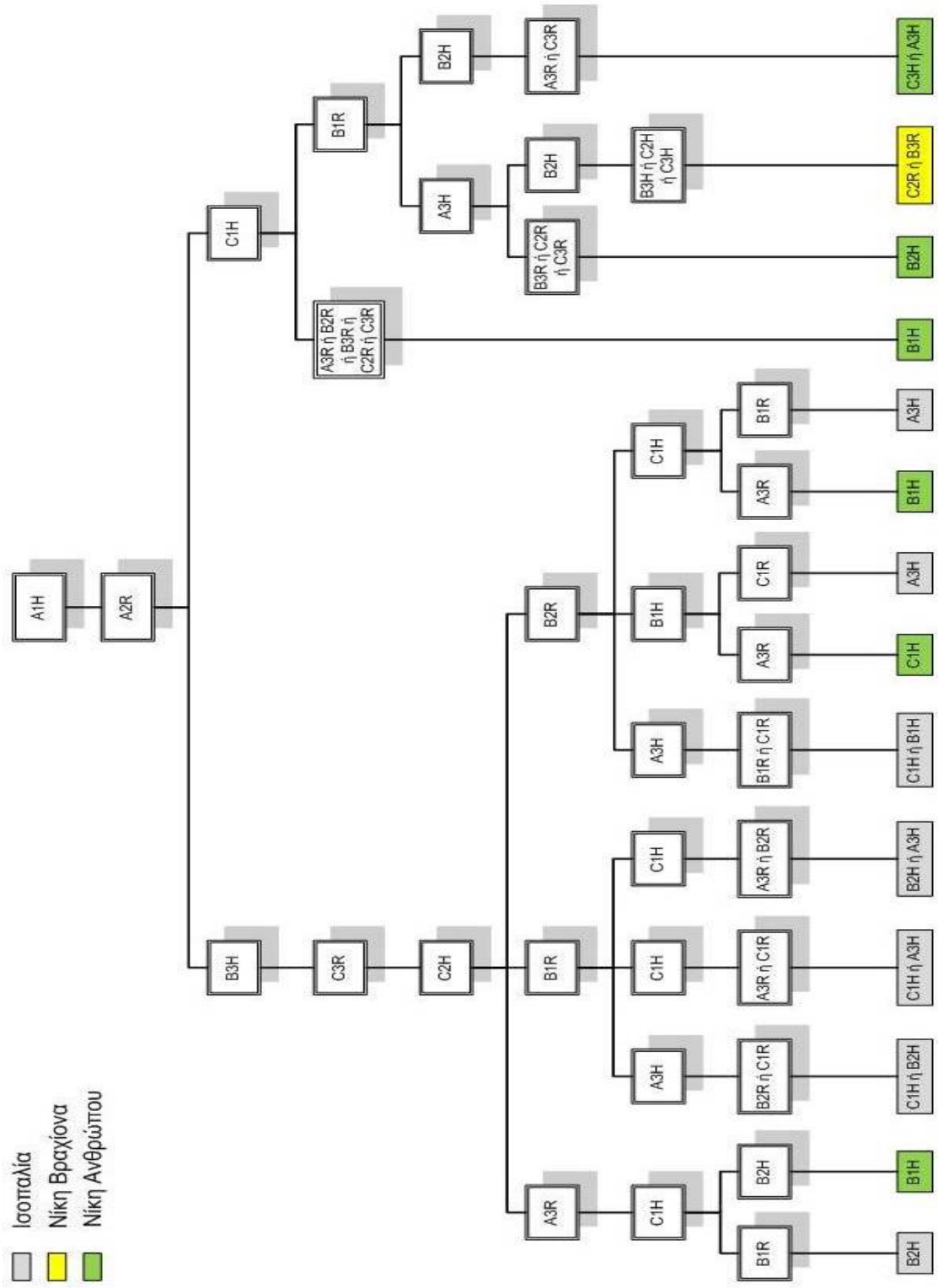
Σχήμα 3.6.3.19


Ισοταλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
C3R
C2R
C1R
B1H
A3H
B2H
C3H
B2R ή C1R
A3R ή C2R
C2R
C1R
B2R
B1R
C1H
B2H
B1R ή C3R
A3R ή C1R
B1R ή C2R
A3R ή B2R
B1H ή B2H ή C1H
C2R
C2R
C2H
C1R
C2R
B2R
C2R
C1R
C1H ή B2H
B2H ή C1H
B2R ή C1R
B2H ή B1H
C1H ή B1H
B2H
C2H
C1H
C2H
C3H ή B1H
B1H ή C3H
B2H ή A3H
B2H ή A3H

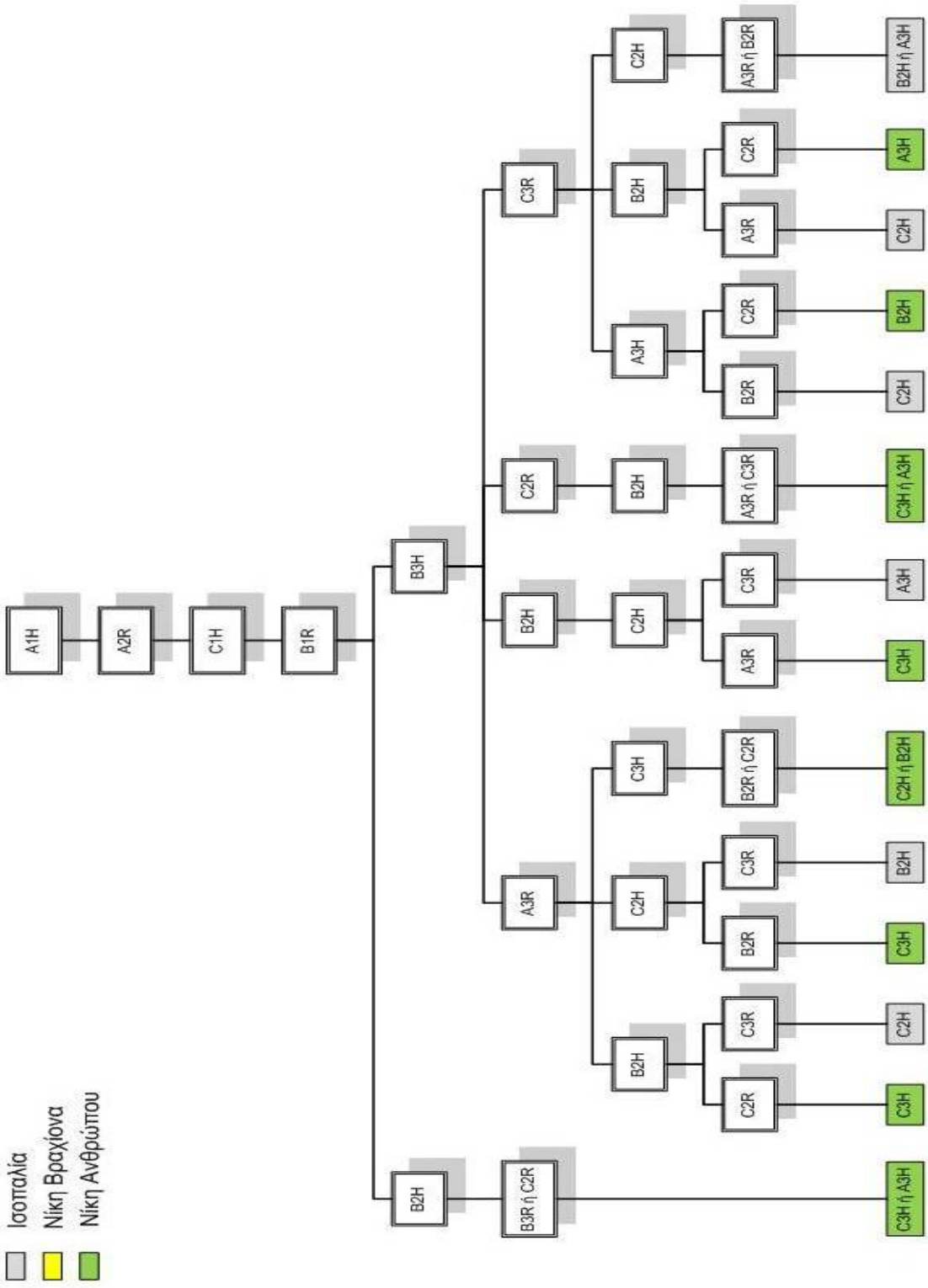
Σχήμα 3.6.3.20


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
C3R
C1H
C2H
A3R
B2H
C1R
B1H
B1R
C1H
B2R ή C1R
C1H ή B2H
B1H
A3R ή B2R ή C2R
C2H
A3R ή B2R
B2H ή A3H
B2H
C2R
A3H
A3R
C2H
A3H
C2R
B2H
B2R
C2H
B1H
B2H
B1R
C2H
A3R ή C1R
C1H ή A3H
C1H
C2R
A3H
A3R
C2H
A3H
C2R
C1H
C1R
C2H
B1H
A3R ή C1R ή C2R

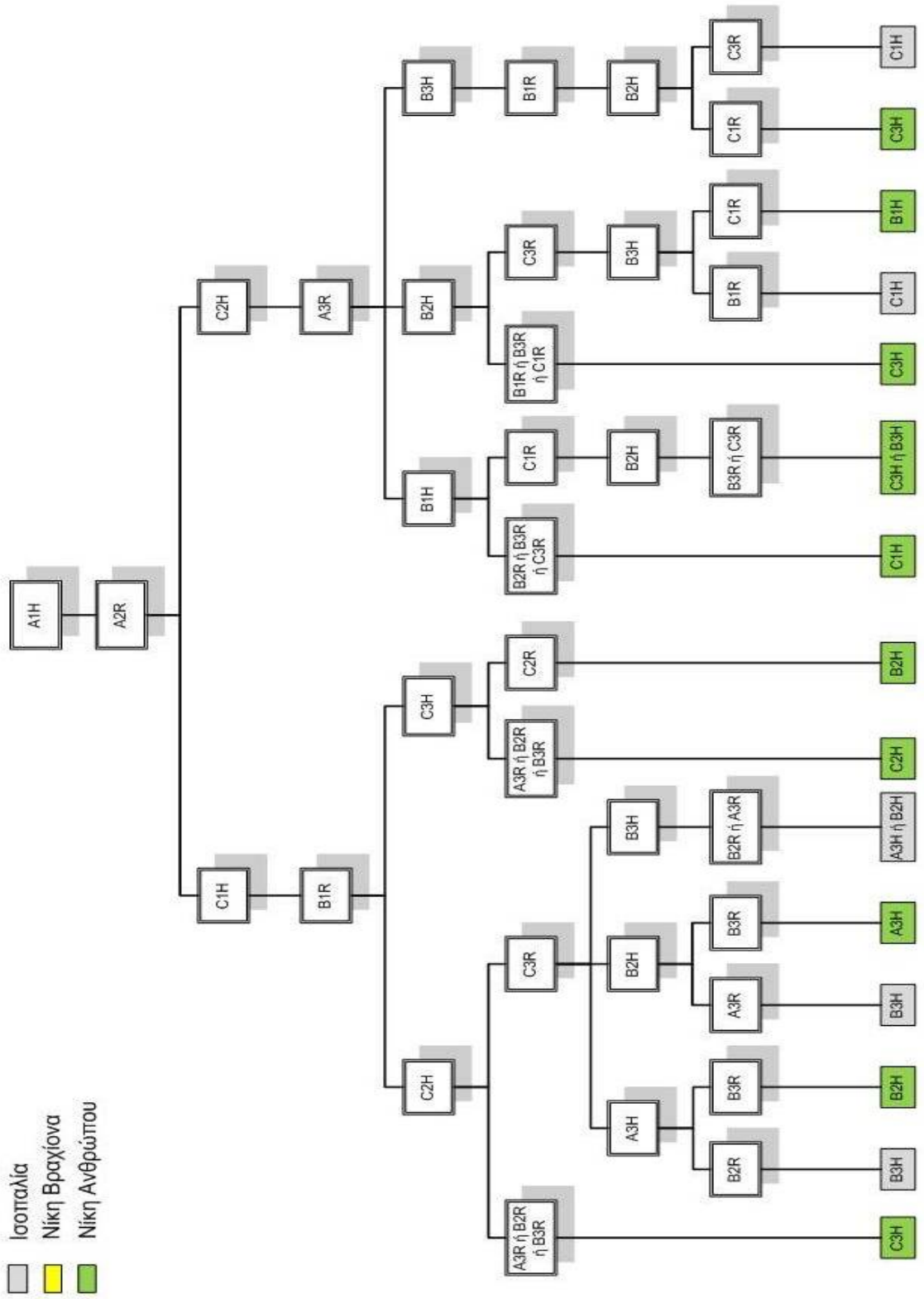
Σχήμα 3.6.3.21


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
B3H
C3R
C2H
C1H
B1R
B2H
A3R ή C3R
C3H ή A3H
A3H
B2H
B3H ή C2H ή C3H
C2R ή B3R
B3R ή C2R ή C3R
B2H
A3R ή B2R ή B3R ή C2R ή C3R
B1H
B2R
B1H
C1H
B1R
A3H
A3R
B1H
C1R
A3H
A3R
C1H
A3H
B1R ή C1R
C1H ή B1H
B1R
C1H
A3R ή B2R
B2H ή A3H
C1H
A3R ή C1R
C1H ή A3H
A3H
B2R ή C1R
C1H ή B2H
A3R
C1H
B2H
B1H
B1R
B2H

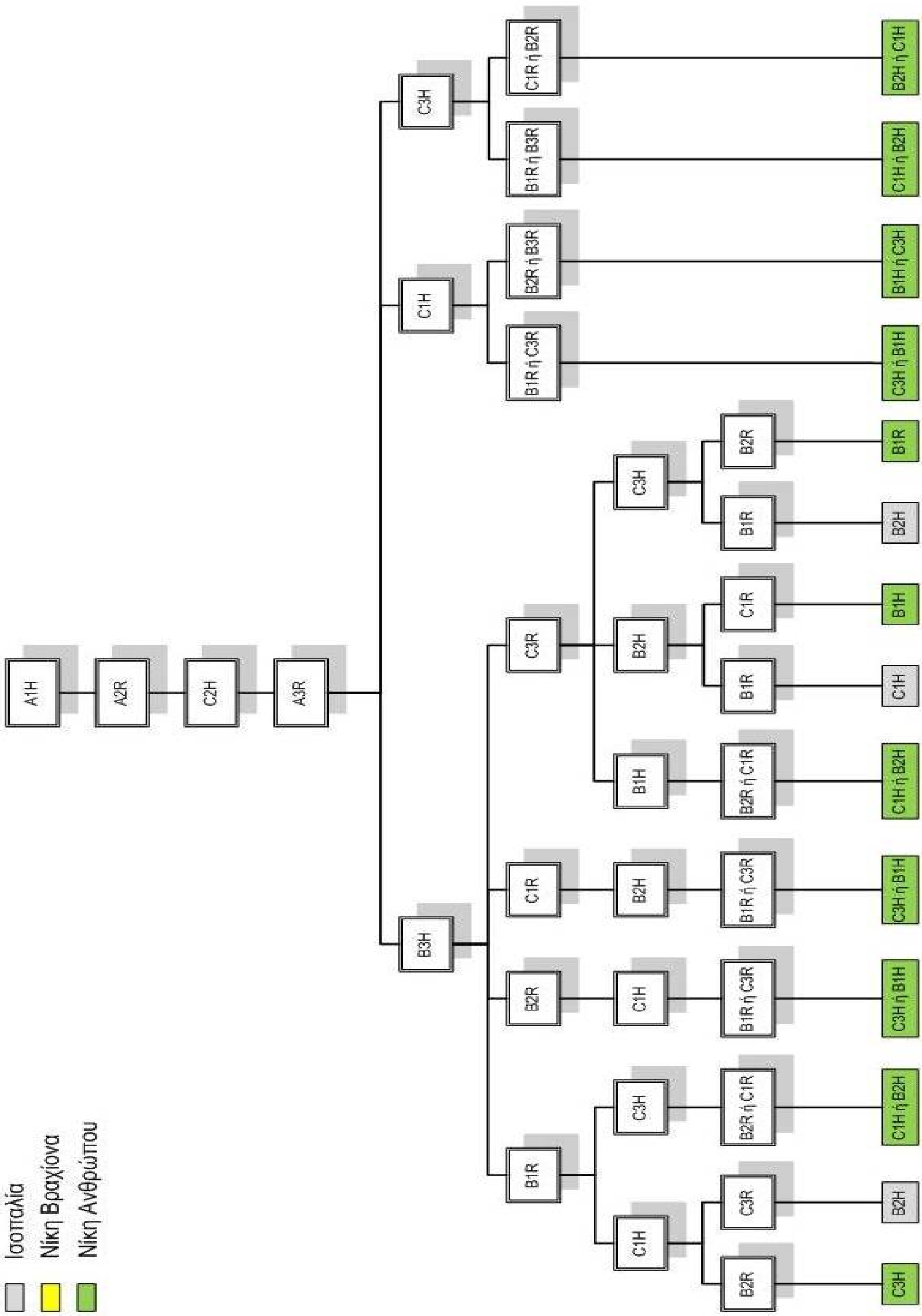
Σχήμα 3.6.3.22


A1H
A2R
C1H
B1R
B3H
B2H
C3R
C2R
B2H
A3R
B3R ή C2R
C2H
B2H
A3H
A3R ή B2R
C2R
A3R
C2R
B2R
A3R ή C3R
C3R
A3R
C3H
C2H
B2H
B2R ή C2R
C3R
B2R
C3R
C2R
C2H ή B2H
B2H
C3H
C2H
C3H
C3H ή A3H
B2H ή A3H
A3H
C2H
B2H
C2H
C3H ή A3H
A3H
C3H
Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου

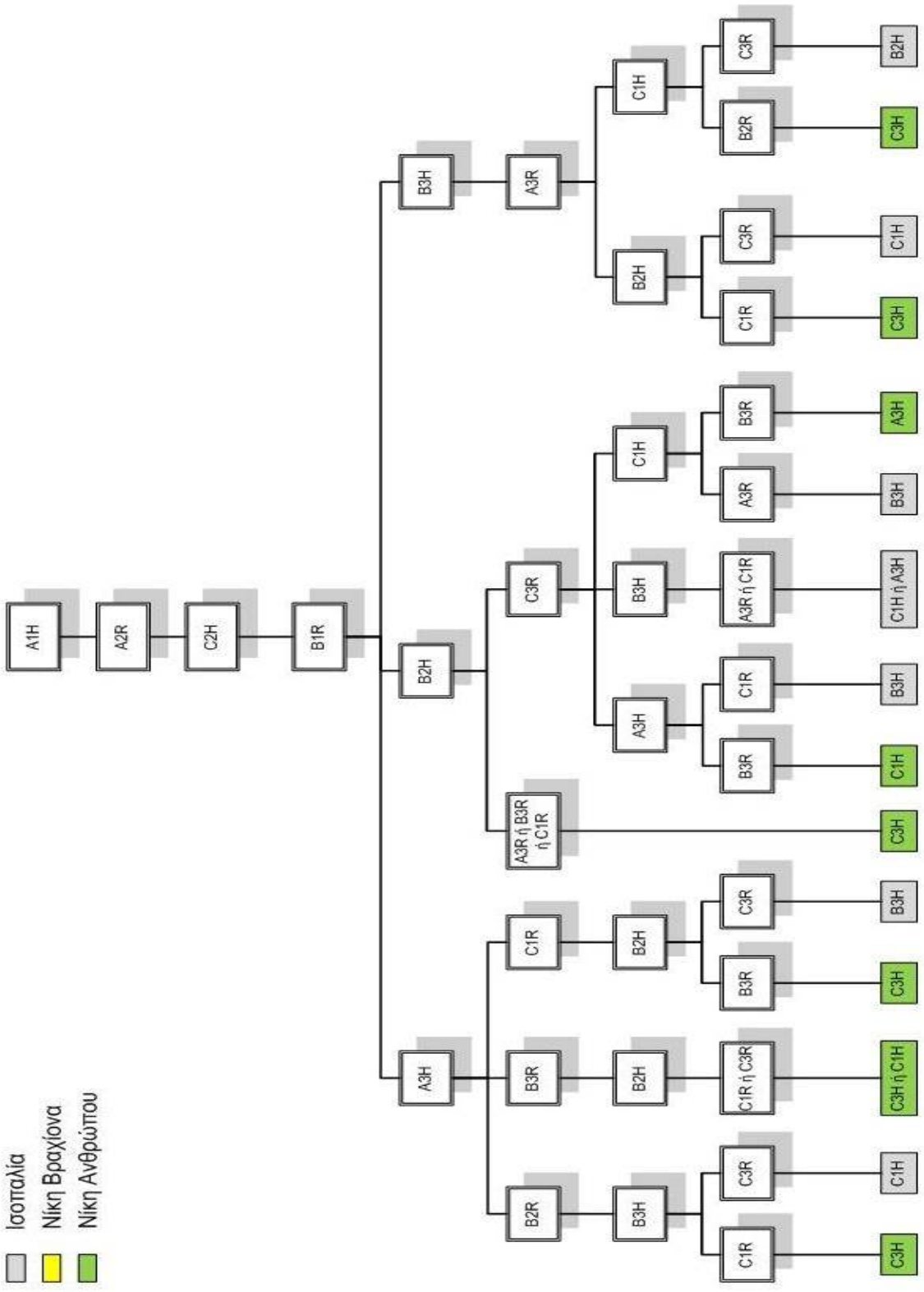
Σχήμα 3.6.3.23


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C1H
A3R
B1R
B3H
B2H
B1H
C3H
C2H
C1H
C3R
B1R
B2H
C1R
B3H
B1R
C3R
B1R ή B3R ή C1R
C1R
B2H
B2R ή B3R ή C1R
B2R ή B3R ή C3R
C2R
A3R ή B2R ή B3R
C3R
B3H
B2H
A3H
B2R ή A3R
B3R
A3R
B3R
B2R
A3R ή B2R ή B3R
C1H
C3H
B1H
C1H
C3H
C3H ή B3H
C1H
B2H
C2H
A3H ή B2H
A3H
B3H
B2H
B3H
C3H

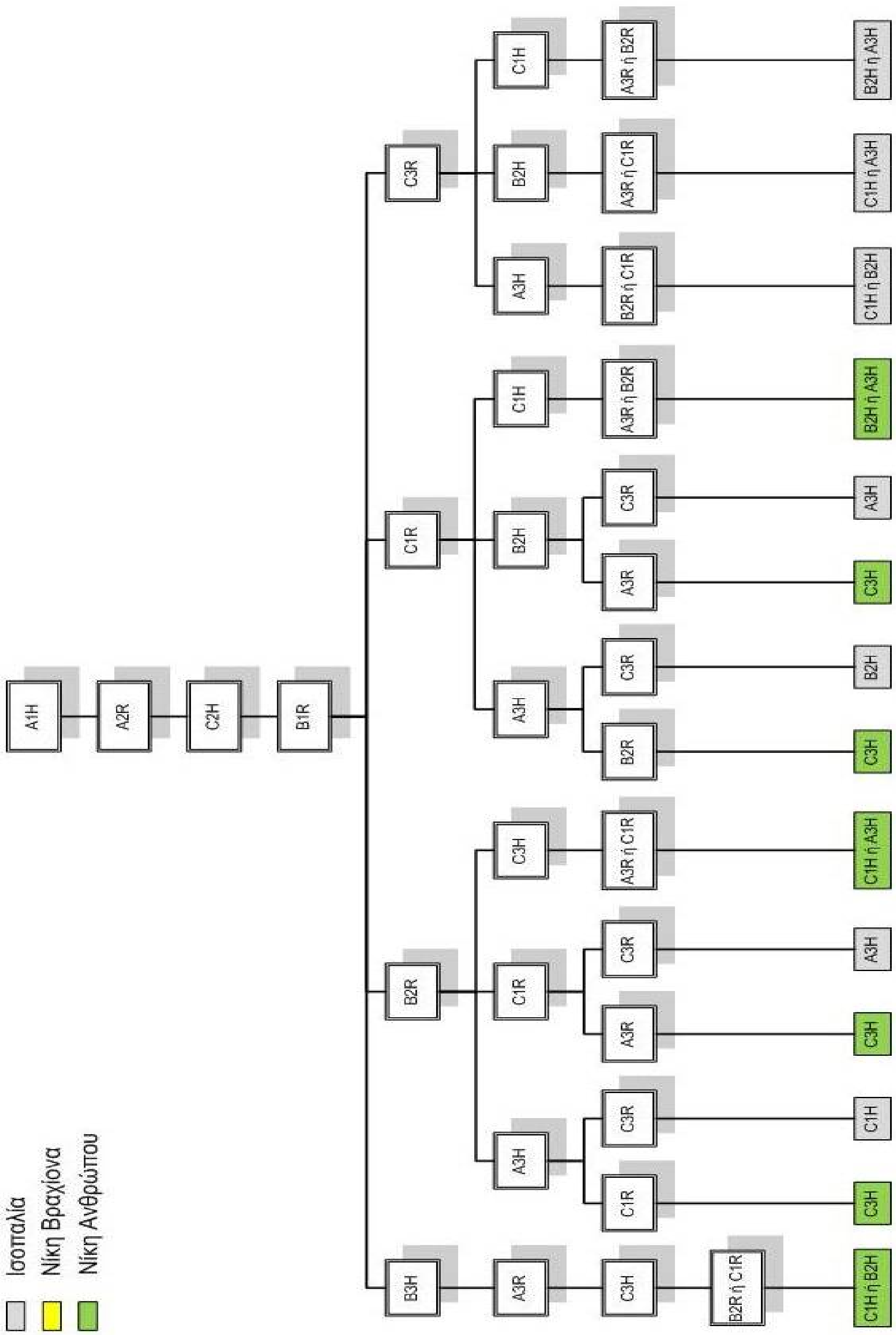
Σχήμα 3.6.3.24


A1H
A2R
C2H
A3R
C3H
C1H
B3H
C1R ή B2R
B1R ή B3R
B2R ή B3R
B1R ή C3R
B2H ή C1H
C1H ή B2H
B1H ή C3H
C3H ή B1H
C3R
C1R
B2R
B1R
C3H
B2H
B1H
C1H
B2R
B1R
C1R
B1R
B2R ή C1R
B1R ή C3R
B1R ή C3R
B2R ή C1R
C3R
B2R
B1R
B2H
B1H
C1H
C1H ή B2H
C3H ή B1H
C3H ή B1H
C1H ή B2H
B2H
C3H
Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου

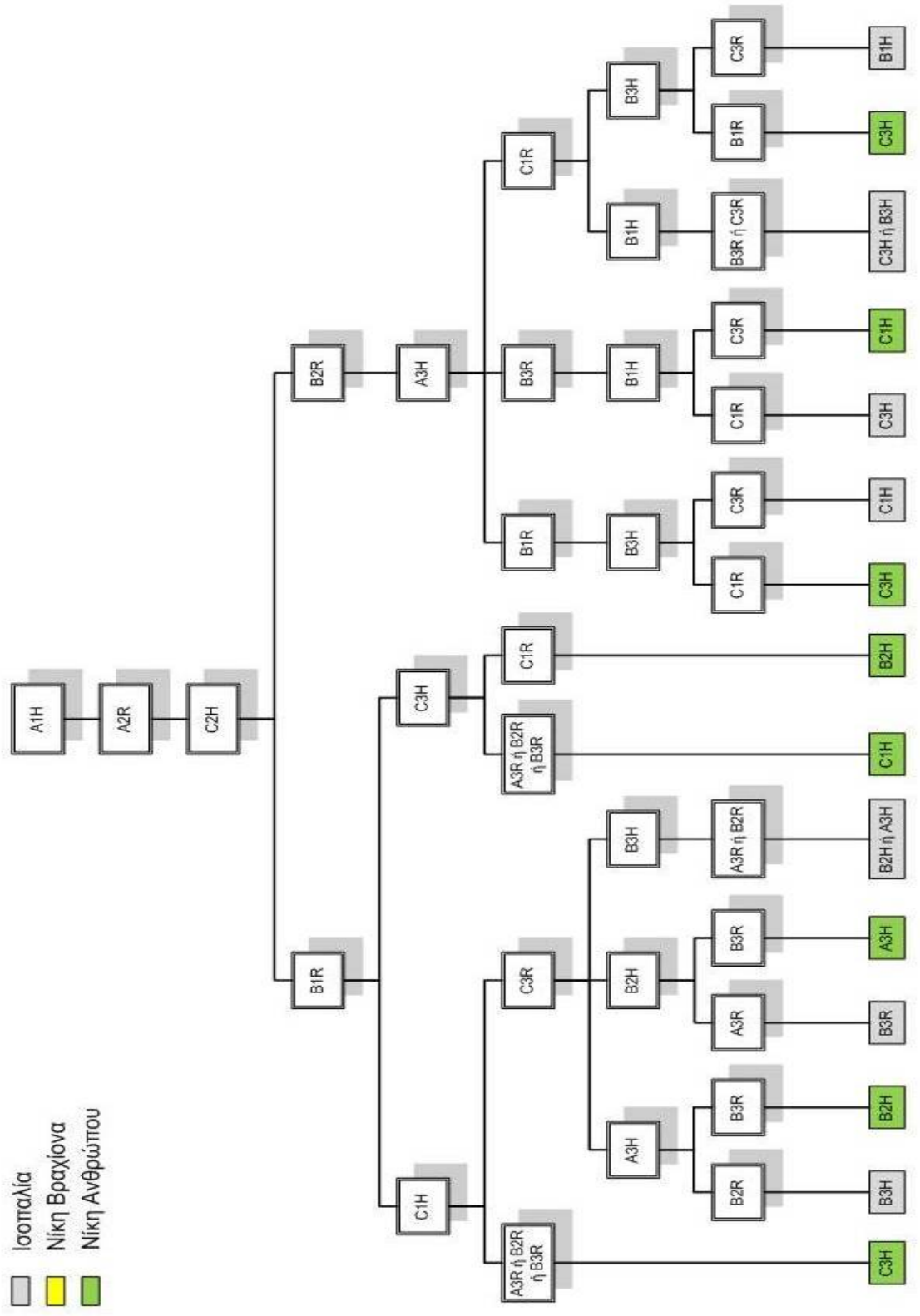
Σχήμα 3.6.3.25


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
B1R
B3H
A3R
C1H
C3R
B2H
C3H
B2R
C1H
C3R
B2H
C1R
C3H
B2H
A3H
C1H
B3R
A3H
B3H
A3R
A3R ή C1R
C1H ή A3H
B3H
C3R
C1R
A3H
B3R
C1H
A3R ή B3R ή C1R
C3H
A3H
C1R
B2H
C3R
B3H
B3R
C3H
B3R
B2H
C1R ή C3R
C3H ή C1H
B2R
B3H
C3R
C1H
C1R
C3H

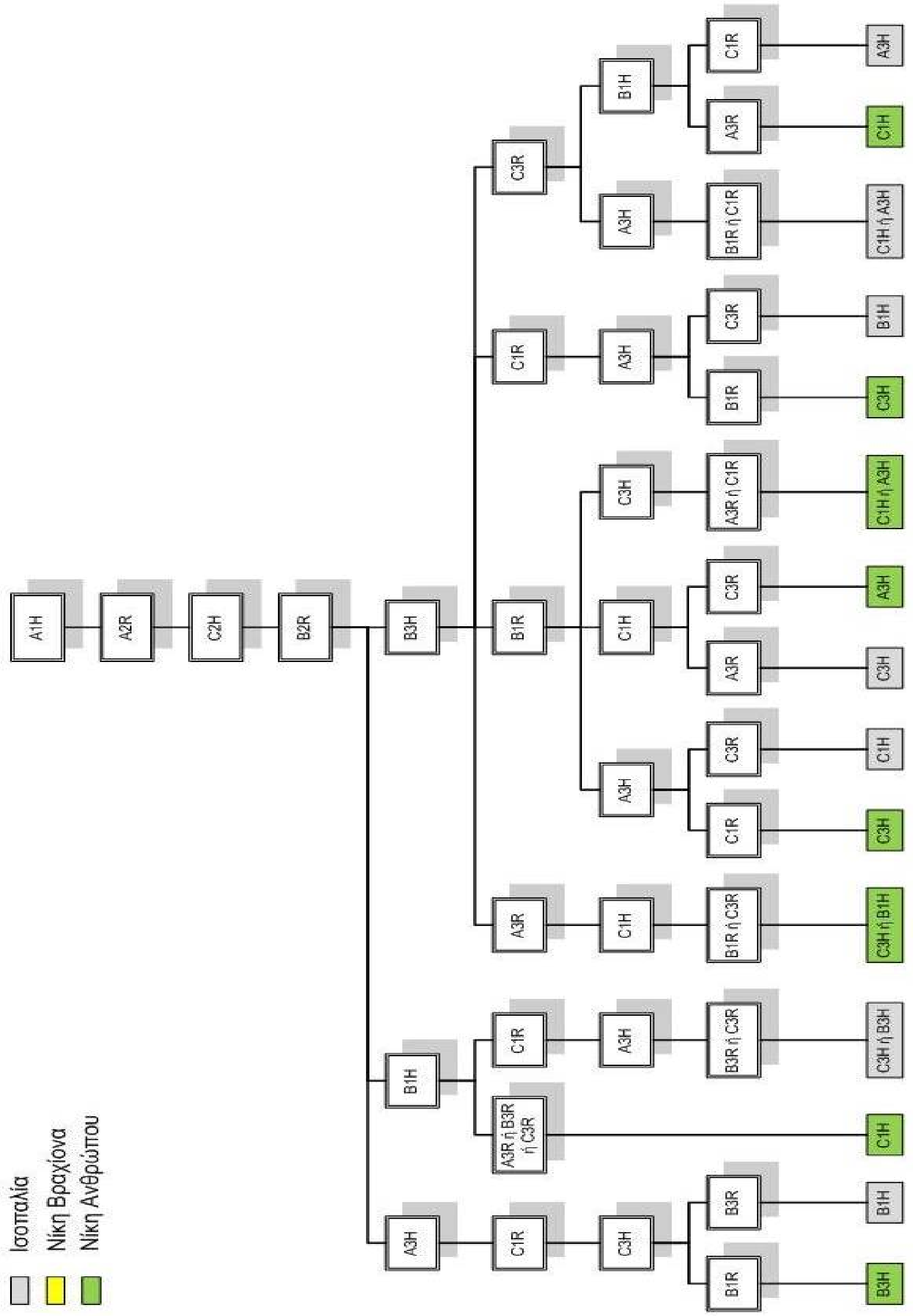
Σχήμα 3.6.3.26


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
B1R
B3H
B2R
C1R
C3R
A3R
C3H
B2R ή C1R
C1H ή B2H
A3H
C1R
A3R
C3R
A3H
C1H
A3R ή C1R
C1H ή A3H
B2R
B2H
B2H
A3R
A3R ή B2R
B2H ή A3H
C1H
B2R ή C1R
C1H ή B2H
A3R ή C1R
C1H ή A3H

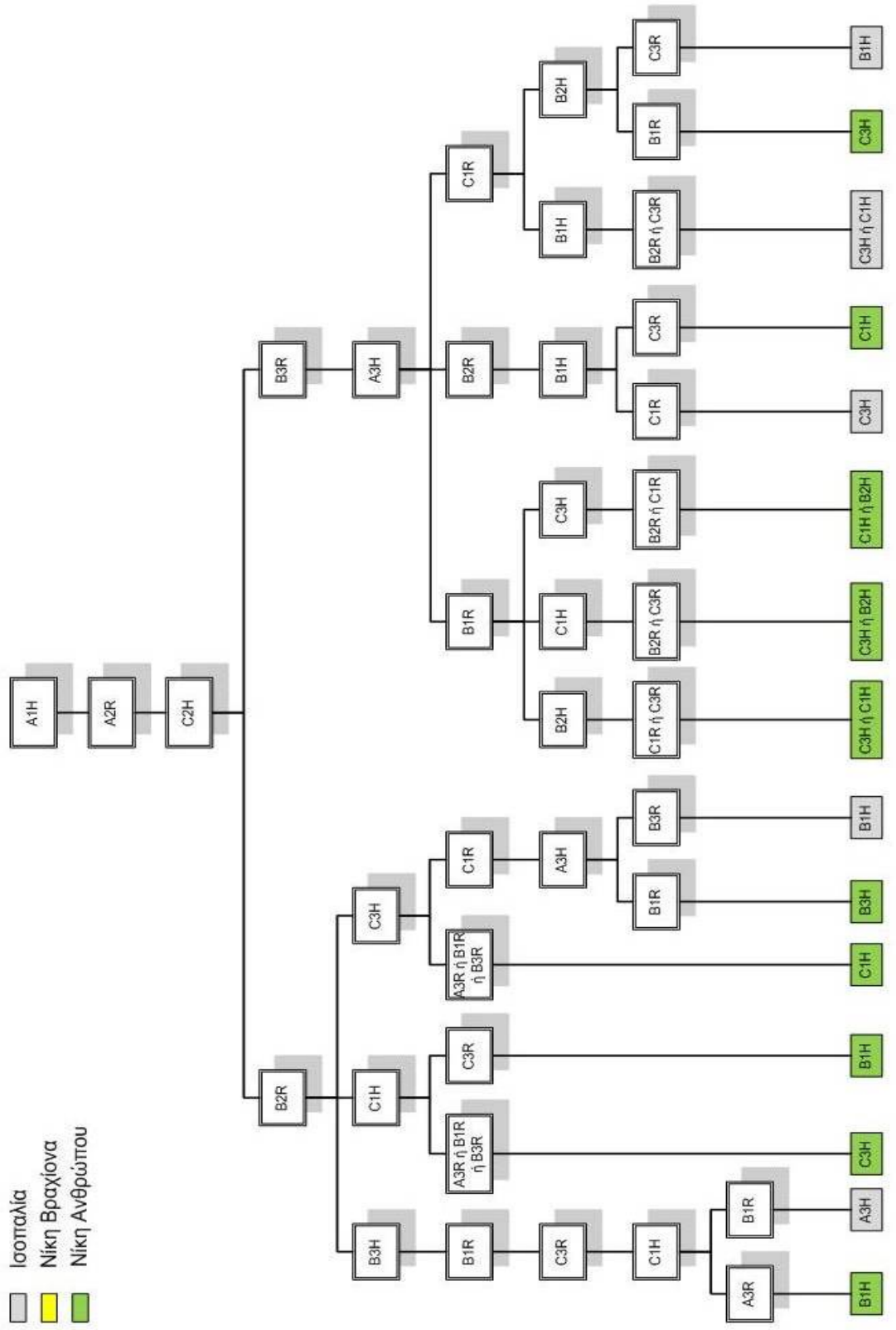
Σχήμα 3.6.3.27


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
B2R
B1R
A3H
C3H
C1H
C1R
B3R
B1R
B1H
C3R
C1R
B3H
B1H
C3R
B1R
B3H
B3R ή C3R
B1H
C3H
C3H ή B3H
C3R
C1R
B3H
B1H
C1H
C3H
C3R
C1R
B3H
B1R
C1H
C3H
C1R
A3R ή B2R ή B3R
B2H
C1H
B3H
B2H
A3H
C3R
A3R ή B2R
B3R
A3R
B2R
B2H ή A3H
A3H
B3R
B2H
B3H
C3H

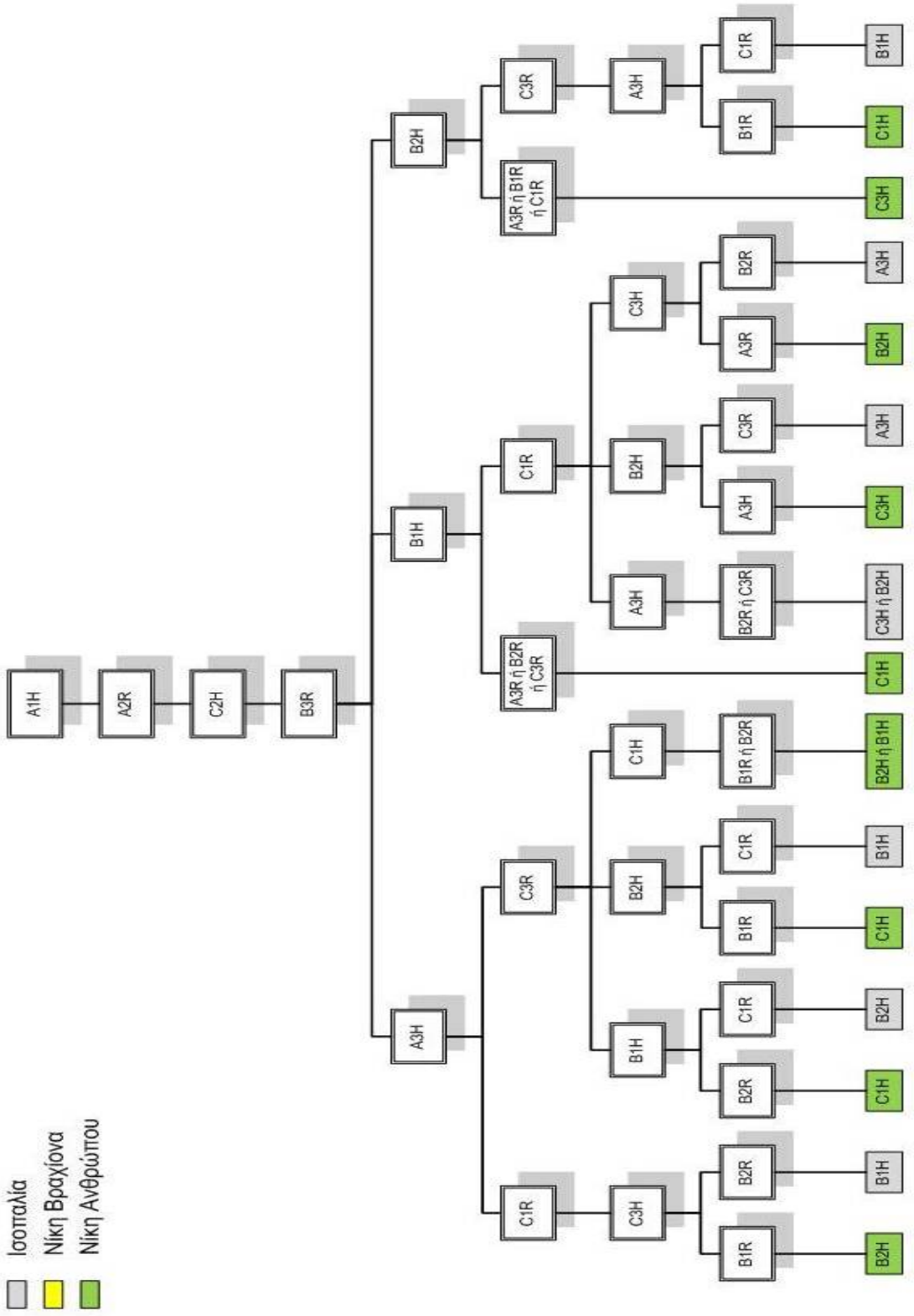
Σχήμα 3.6.3.28


A1H
A2R
C2H
B2R
B3H
B1H
A3H
C3R
C1R
B1R
A3R
A3R ή B3R ή C3R
B1H
A3H
C3H
C1H
C1R
B3R
B1R
B1R ή C1R
A3R ή C1R
B1R ή C3R
B3R ή C3R
C1H ή A3H
C3H ή B1H
C3H ή B3H
B1H
B3H
Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου

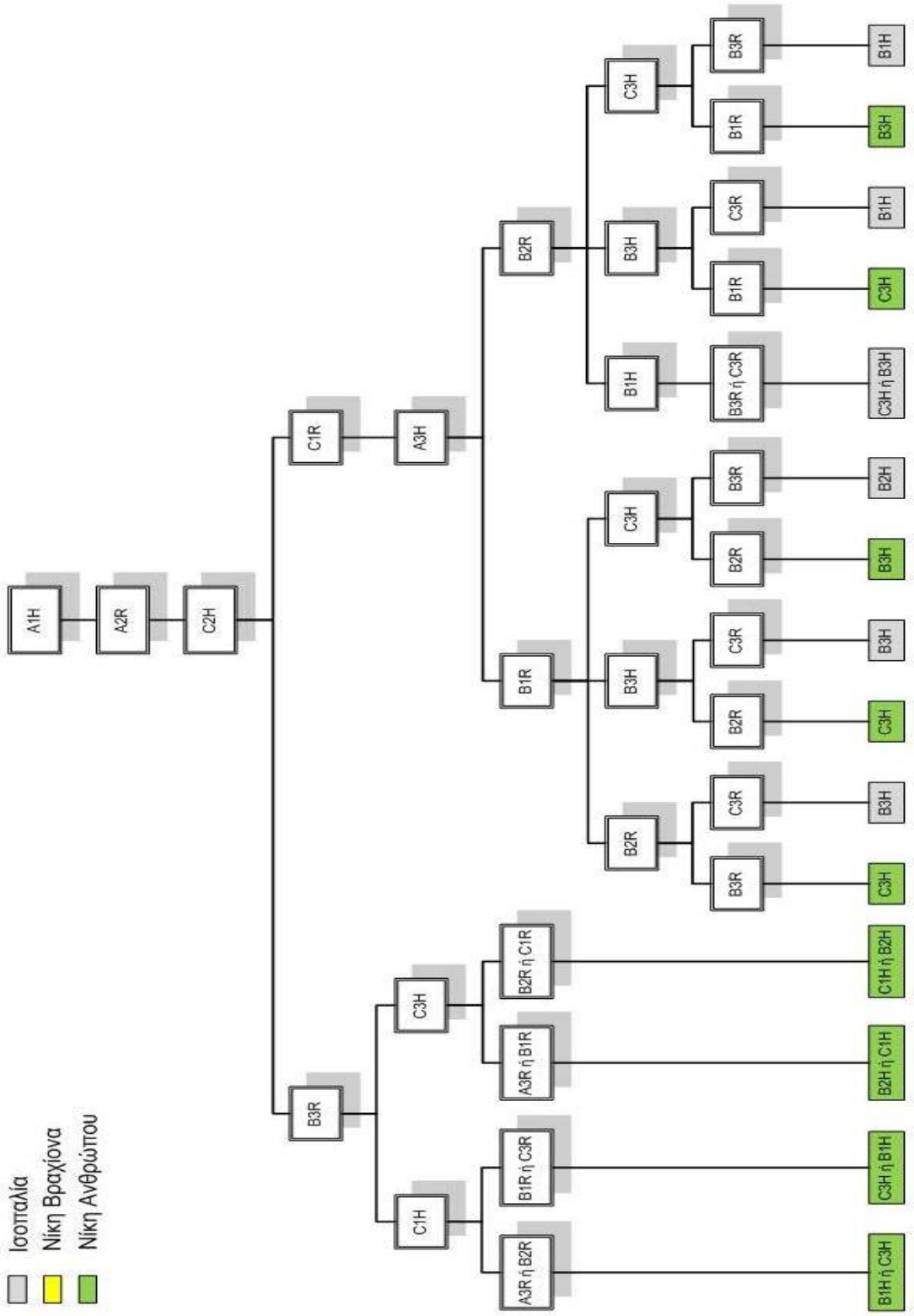
Σχήμα 3.6.3.29


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
B3R
B2R
A3H
C3H
C1H
B3H
B1R
C3R
C1H
A3R
B1H
C3H
B1H
C1R
B2R
A3H
C3H
B1R
A3R ή B1R ή B3R
B3R
B1R
B1H
B3H
C1H
B1H
C3H
C1R
B2H
B1H
C3R
B1R
B2R ή C3R
B1H
C3H
C3H ή C1H
C1R
B2R
B1H
C3R
C1R
C1H
C3H
B2R ή C1R
C1H
B1R
C1H
B2R ή C3R
C3H ή B2H
B2H
C1R ή C3R
C3H ή C1H

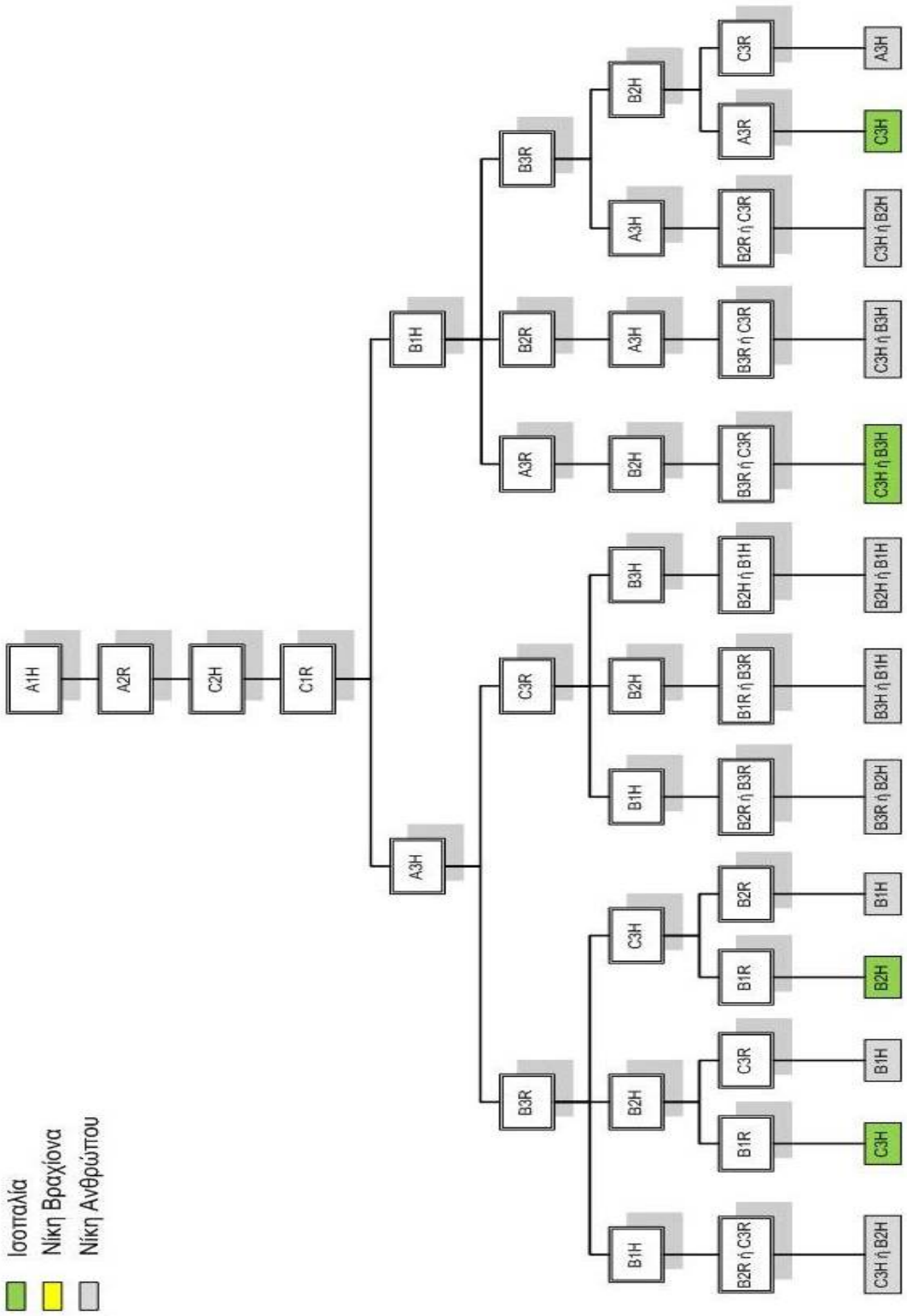
Σχήμα 3.6.3.30


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
B3R
B2H
B1H
A3H
C3R
A3H
C1R
B1R
B1H
C1H
A3R ή B1R ή C1R
C3H
C1R
B2H
C3H
B2R
A3R
A3H
B2H
B2H
C3R
A3H
A3H
C3H
A3H
B2R ή C3R
C3H ή B2H
A3R ή B2R ή C3R
C1H
C1H
B1R ή B2R
B2H ή B1H
C3R
B2H
C1R
B1R
B1H
C1H
B1H
C1R
B2R
B2H
C1H
C3H
C1R
B2R
B1H
B1R
B2H

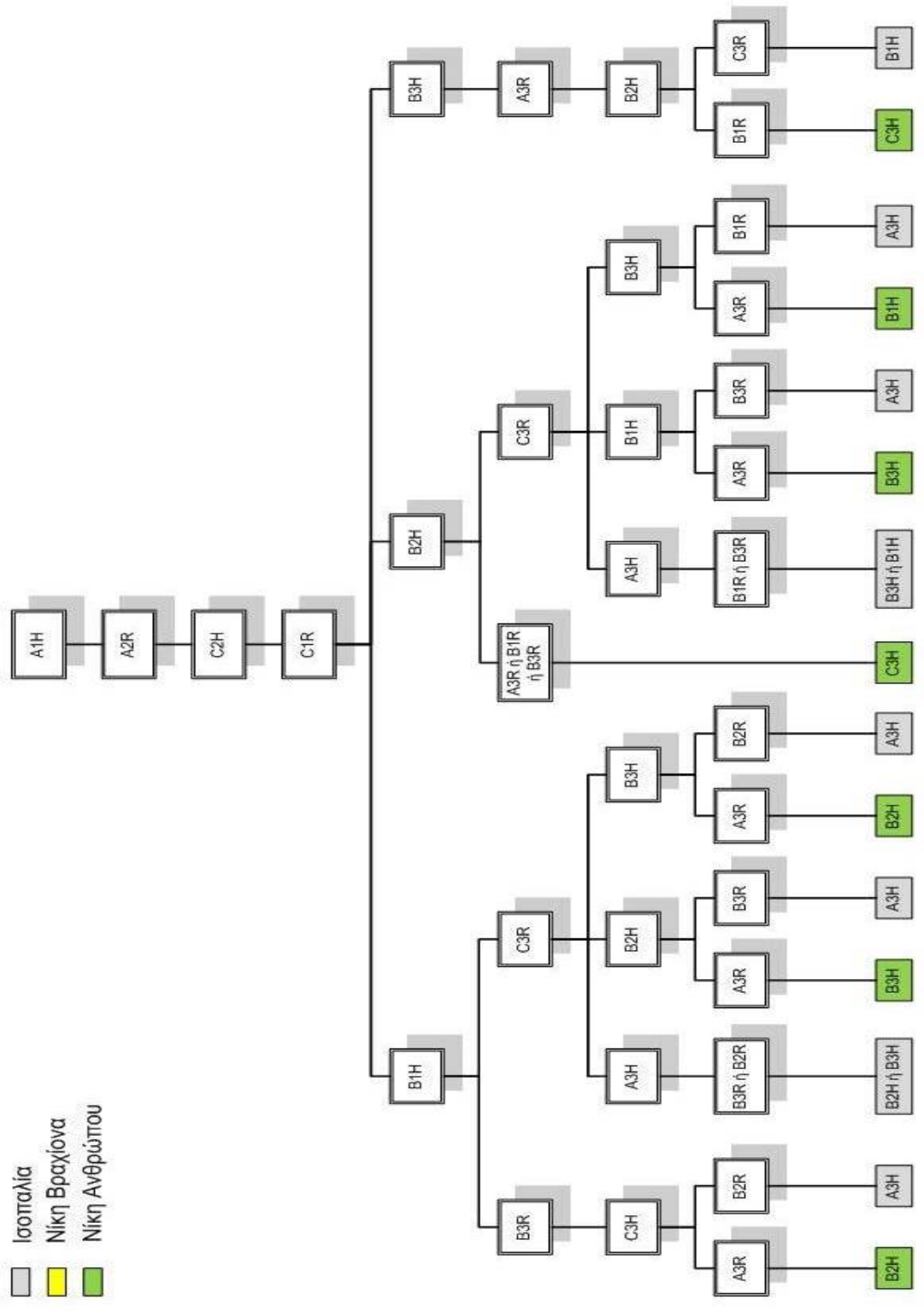
Σχήμα 3.6.3.31


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C1R
A3H
B2R
B1R
C3H
B3H
B2R
B3R
C3R
B2R
B1H
B3R ή C3R
B1R
C3R
B1R
B3R
B1R
C3H
B3H
B2R
C3R
B2R
B3R
C3R
B2R
B1H
B3H
C3H
B3H
B2H
B3H
B3H
C3H
B3H
C3H
C3H ή B3H
C1H ή B2H
B2H ή C1H
C3H ή B1H
B1H ή C3H
B3R
C3H
C1H
B2R ή C1R
A3R ή B1R
B1R ή C3R
A3R ή B2R

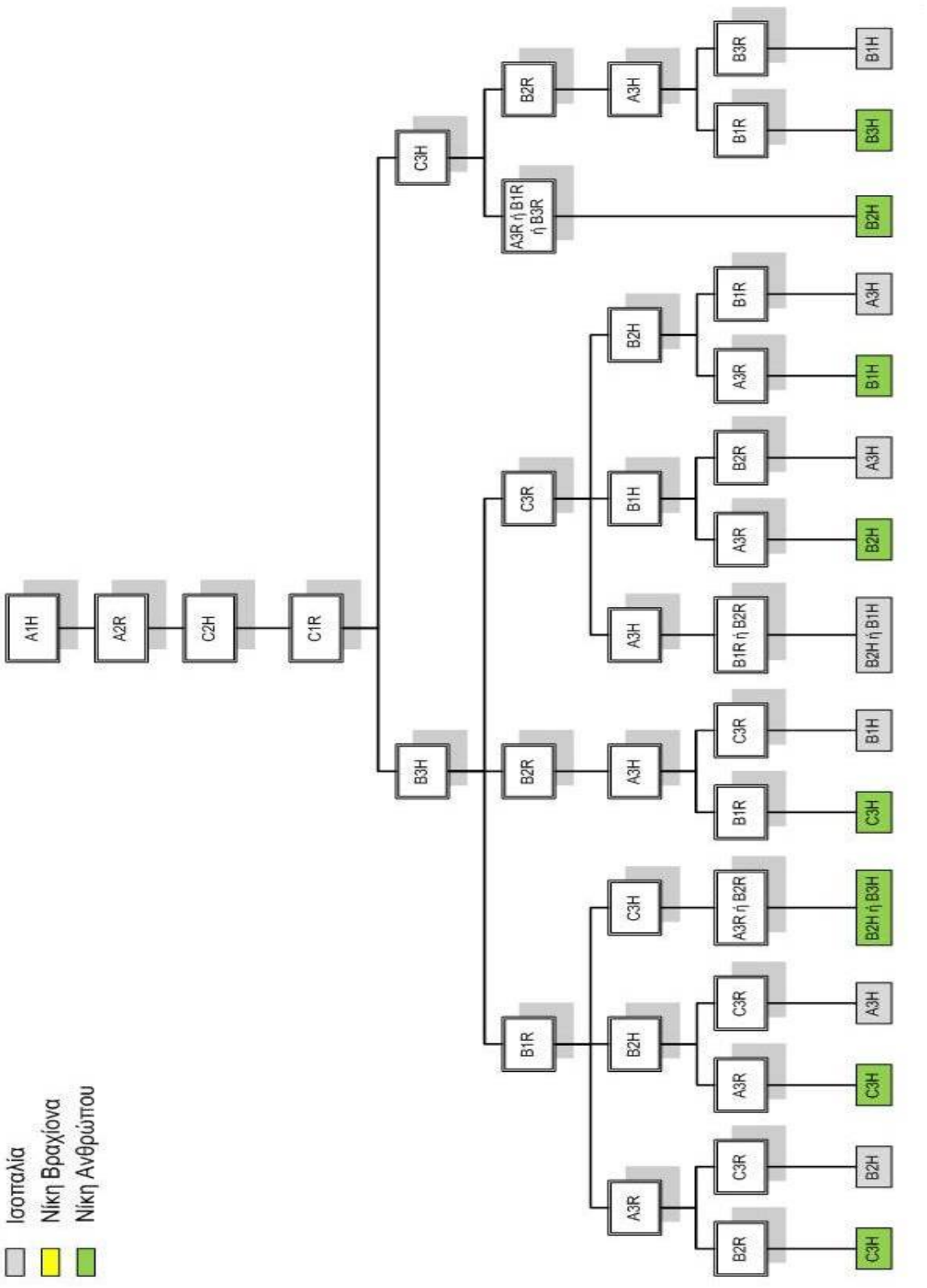
Σχήμα 3.6.3.32


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C1R
B1H
A3H
B3R
B2R
A3R
C3R
B2H
A3H
B1H
B3H
B2H
B1H
C3H
B2R
B1R
C3R
B1R
B2H
C3R
B2H
A3H
A3H
B2H
A3H
B2H
B3R ή C3R
B3R ή C3R
B2H ή B1H
B1R ή B3R
B2R ή B3R
C3H
B1H
C3H
B1H
B2H
C3H
B1H
C3H
B2R ή C3R
B2R ή C3R
C3H ή B2H
C3H ή B3H
C3H ή B3H
B2H ή B1H
B3H ή B1H
B3R ή B2H
B1H
B2H
B1H
C3H
C3H ή B2H

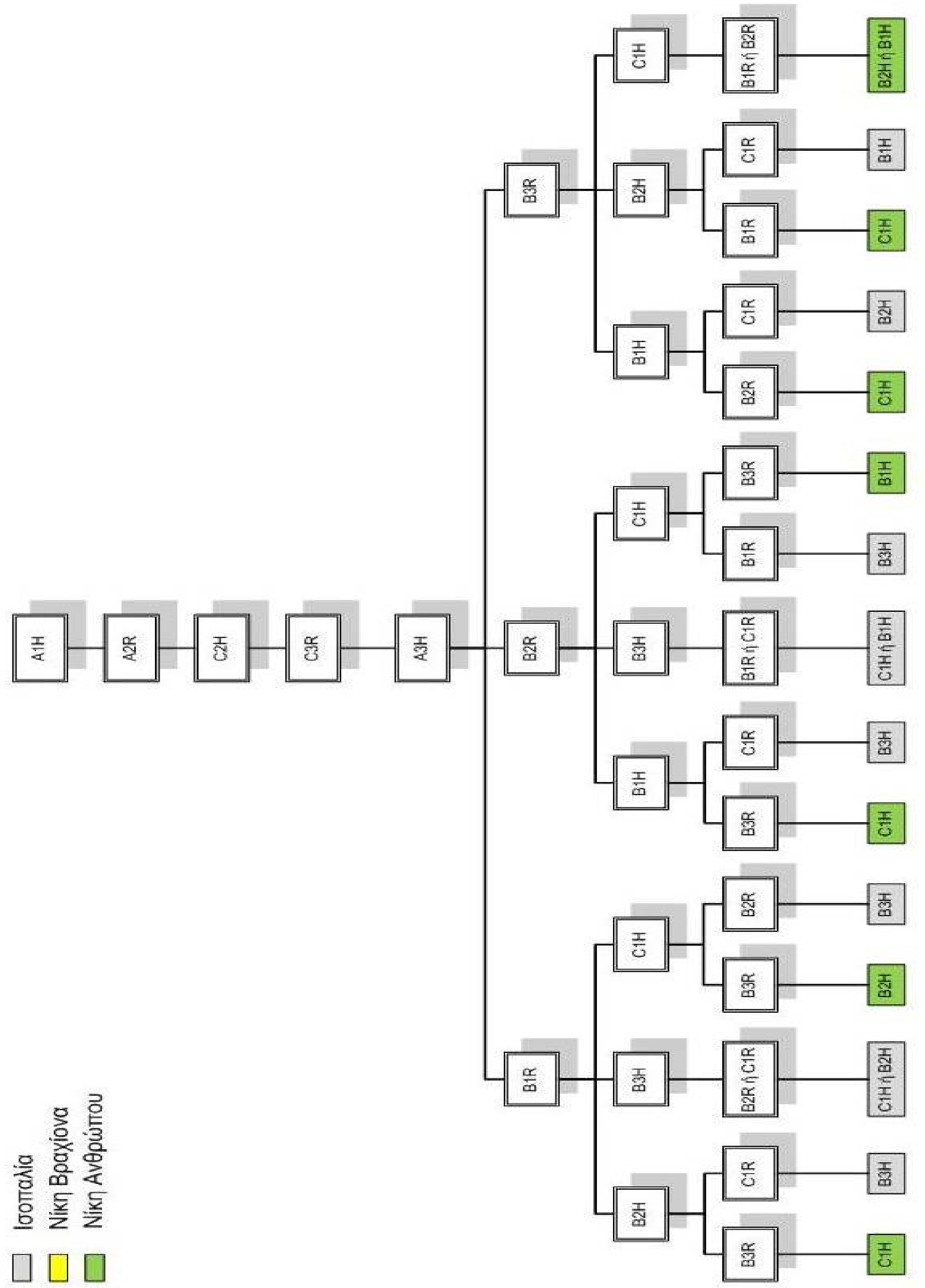
Σχήμα 3.6.3.33


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C1R
B2H
B1H
B3H
A3R
B2H
C3R
B1R
B1H
C3H
C3R
B3H
B1R
A3H
A3R
B1H
B1H
B3R
A3H
A3R
B3H
A3H
B1R ή B3R
B3H ή B1H
A3R ή B1R ή B3R
C3H
C3R
B3H
B2R
A3H
A3R
B2H
B2H
B3R
A3H
A3R
B3H
A3H
B3R ή B2R
B2H ή B3H
B3R
C3H
B2R
A3H
A3R
B2H

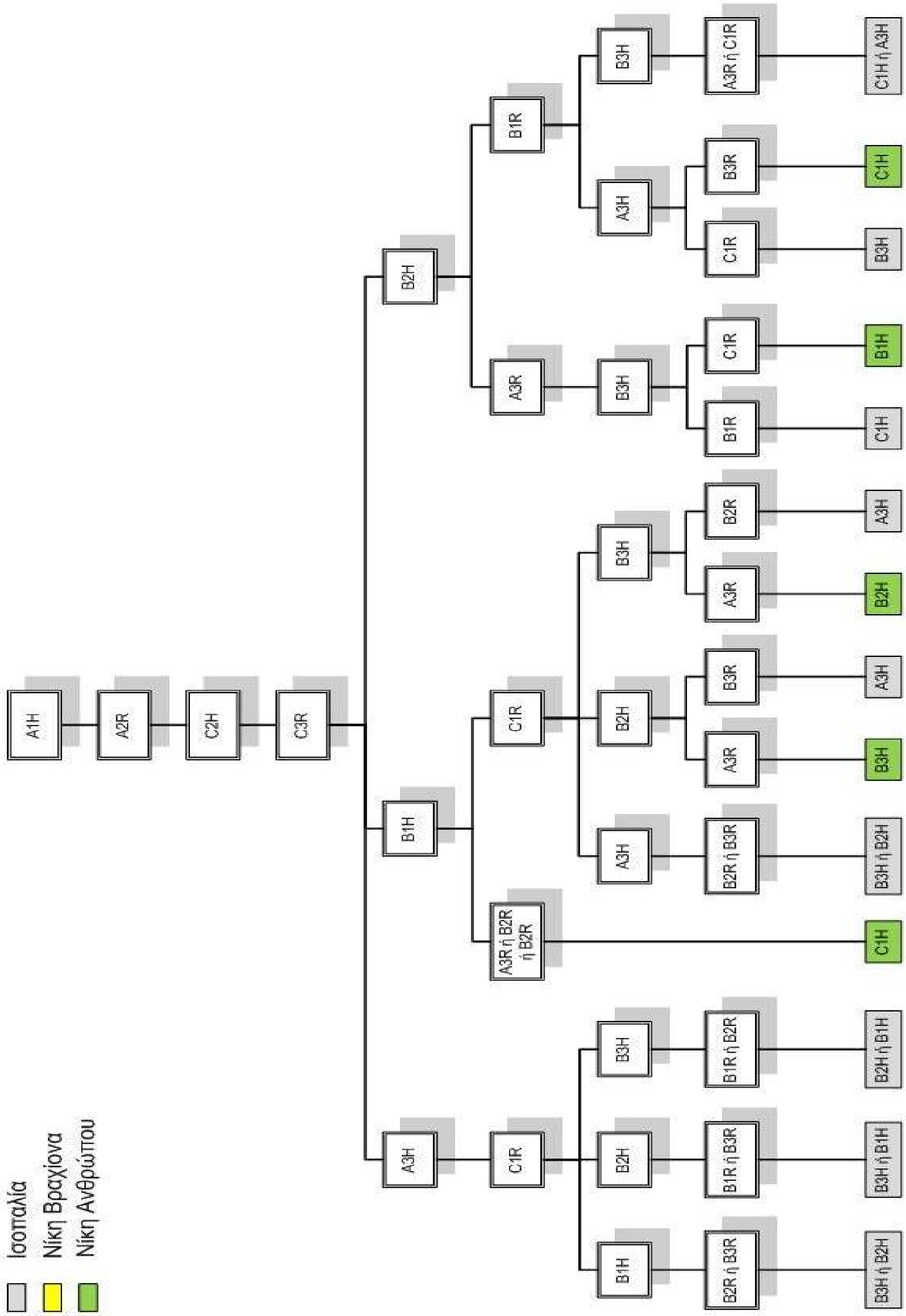
Σχήμα 3.6.3.34


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C1R
B3H
C3H
B2R
A3H
B3R
B1R
B1H
B3H
A3R ή B1R ή B3R
B2H
B2R
A3H
B1R
A3H
A3R
B1H
B1H
C3R
B1H
B2H
A3H
A3R
B2H
A3H
B1R ή B2R
B2H ή B1H
B2R
A3H
C3R
B1H
B1R
C3H
B2H
C3H
A3R ή B2R
B2H ή B3H
C3R
A3H
A3R
C3H
A3R
C3R
B2H
B2R
C3H

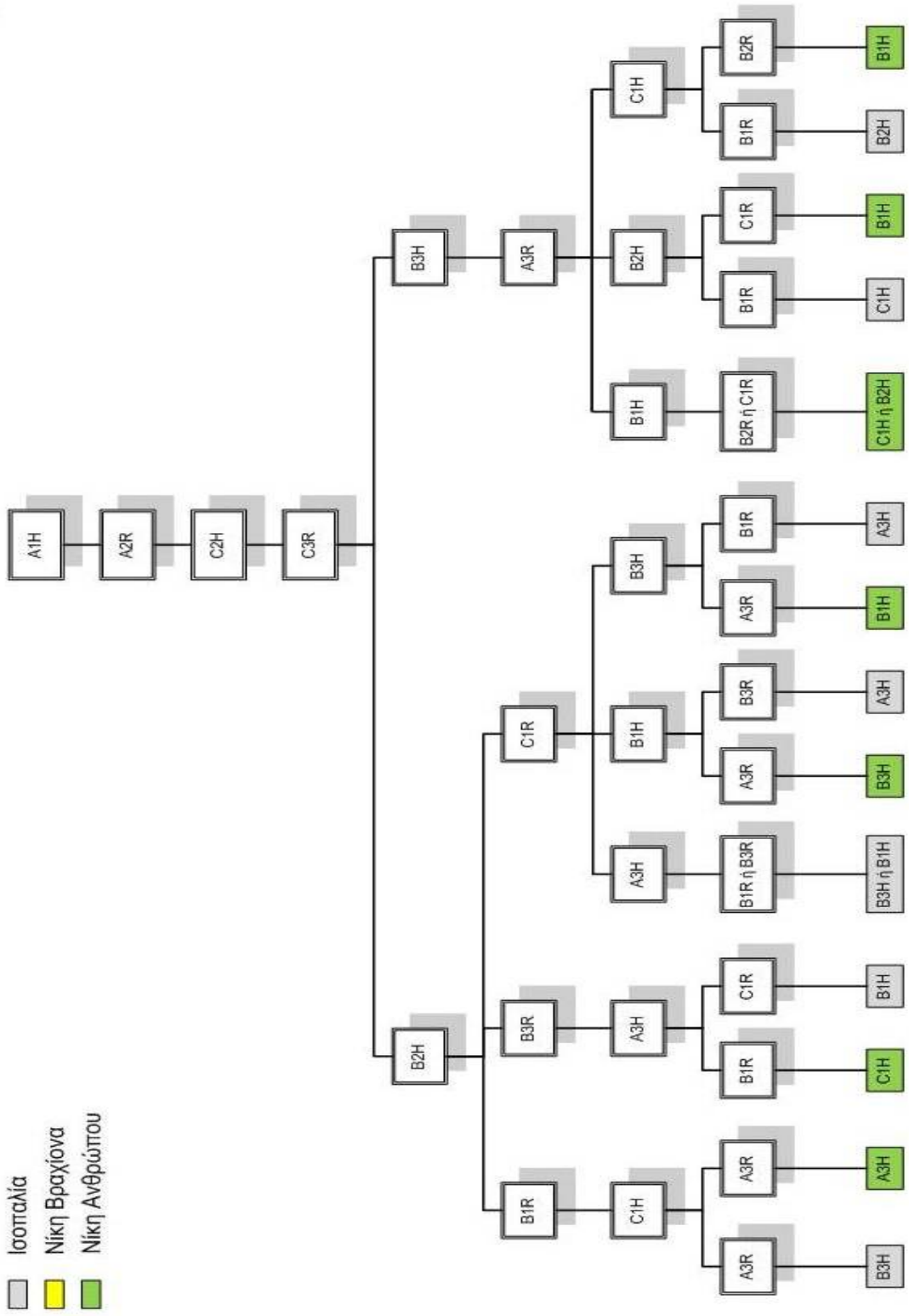
Σχήμα 3.6.3.35



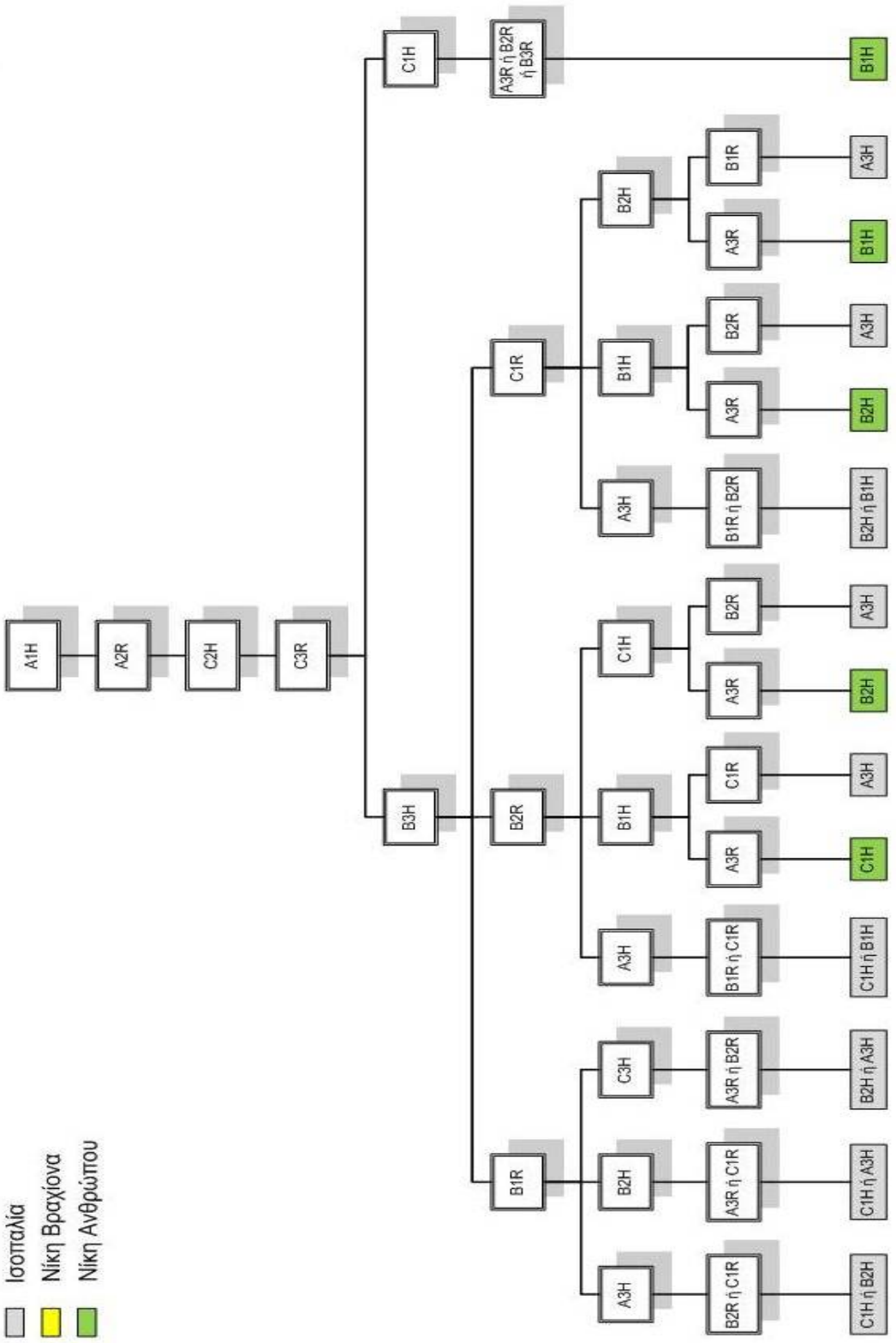
Σχήμα 3.6.3.37


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C3R
A3H
B1H
B2H
C1R
A3R ή B2R ή B2R
A3R
B1R
B1H
B2H
B3H
A3H
B2R ή B3R
B1R ή B3R
B1R ή B2R
B3R
A3R ή C1R
B3H ή B2H
B3H ή B1H
B2H ή B1H
C1H
C1H ή A3H

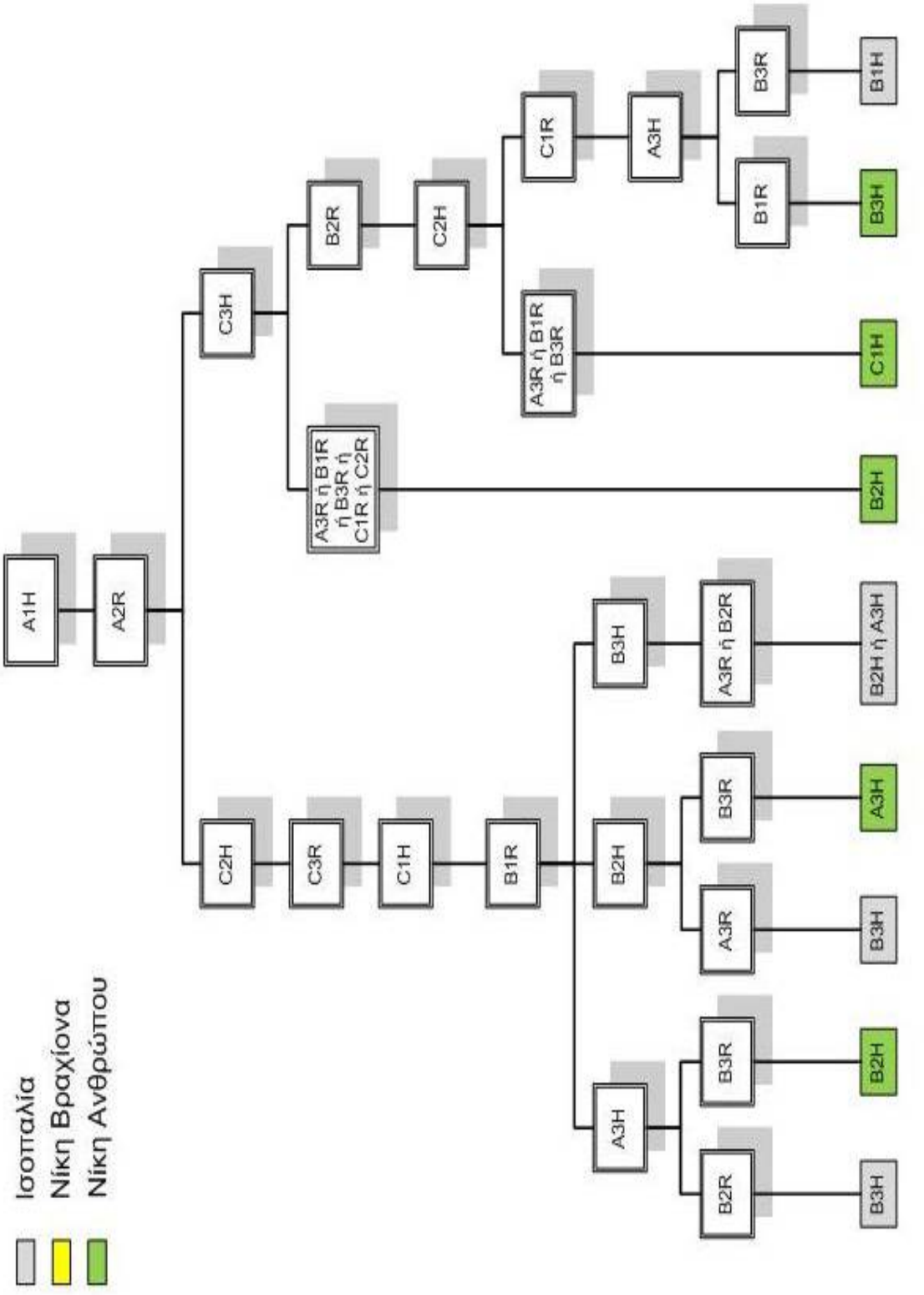
Σχήμα 3.6.3.38


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C3R
B3H
A3R
C1H
B2R
B1H
B1R
B2H
C1R
B1H
B1R
C1H
B2H
B1H
B2R ή C1R
C1H ή B2H
B2H
C1R
B3H
B1R
A3H
A3R
B1H
B1H
B3R
A3H
A3R
B3H
A3H
B1R ή B3R
B3H ή B1H
B3R
A3H
C1R
B1H
B1R
C1H
C1H
A3R
A3H
A3R
B3H

Σχήμα 3.6.3.39


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C2H
C3R
B3H
C1H
A3R ή B2R ή B3R
B1H
B2R
C1R
B1H
A3H
B1R
A3R
B2H
B2R
A3R
B2H
A3H
B1H
A3H
B1R ή B2R
B2H ή B1H
B2R
C1H
A3R
B2H
A3H
C1R
B1H
A3R
C1H
A3H
A3H
B1R ή C1R
C1H ή B1H
C3H
B2H
A3H
A3R ή B2R
A3R ή C1R
B2R ή C1R
B2H ή A3H
C1H ή A3H
C1H ή B2H

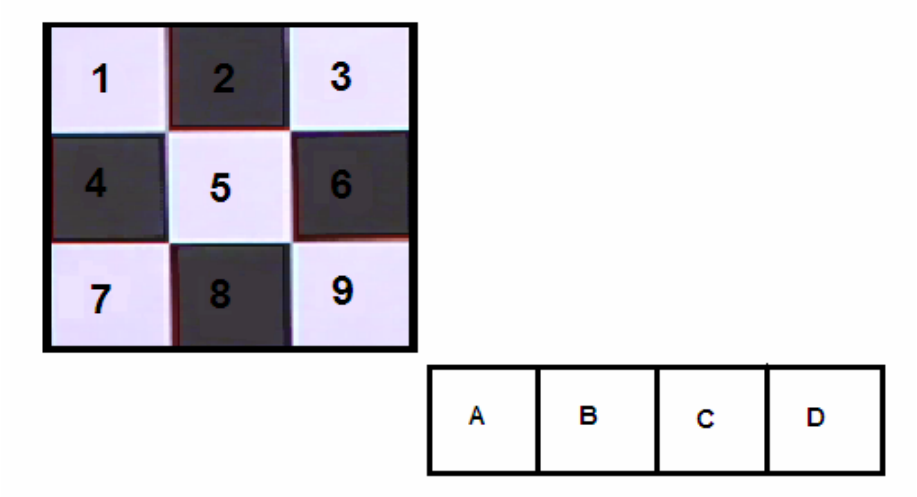
Σχήμα 3.6.3.40


Ισοπαλία
Νίκη Βραχίονα
Νίκη Ανθρώπου
A1H
A2R
C3H
C2H
B2R
C2H
C1R
A3H
B3R
B1H
B1R
B3H
A3R ή B1R ή B3R
C1H
A3R ή B1R ή B3R ή C1R ή C2R
B2H
C2H
C3R
C1H
B1R
B3H
A3R ή B2R
B2H ή A3H
B2H
B3R
A3H
A3R
B3H
A3H
B3R
B2H
B2R
B3H

Σχήμα 3.6.3.41

3.6.3.1 ΚΙΝΗΣΗ ΒΡΑΧΙΟΝΑ

Για να τοποθετήσει ο βραχίονας το πιόνι του στη θέση που επέλεξε πρέπει να γνωρίζει τις συντεταγμένες που αντιστοιχούν σε κάθε κουτί της σκακιέρας. Οι συντεταγμένες αυτές για τα εννιά κουτιά που χρησιμοποιούμε αλλά και για τα πιόνια που παίρνει ο βραχίονας για να τα τοποθετήσει στο κουτί που επέλεξε είναι:



Εικόνα 3.6.3.1.1 Κουτιά που χρησιμοποιούνται από την σκακιέρα

KOYTI 1 : x=0.0533 y=0.2818 z=0.0446	*KOYTI 2 : x=0.0000 y=0.2817 z=0.0423*	*KOYTI 3 : x=-0.035 y=0.2807 z=0.0423*
KOYTI 4 : x=0.0640 y=0.3256 z=0.0375	*KOYTI 5 : x= 0.0000 y=0.3247 z=0.0407*	*KOYTI 6 : x=-0.0389 y=0.3172 z=0.0381*
KOYTI 7 : x=0.0652 y=0.3698 z=0.0362	*KOYTI 8 : x=0.0100 y= 0.3680 z= 0.0390*	*KOYTI 9: x=-0.042 y=0.3629 z=0.0310*

KOYTI A : x=0.1380 y=0.0100 z=0.0130	*KOYTI B : x=0.1850 y=0.0100 z=0.0080*	*KOYTI C : x=0.2294 y=0.0100 z=0.0027*	*KOYTI D : x=0.2867 y=0.0100 z=0.0000*

Πίνακας 3.6.3.1.1 Συντεταγμένες κουτιών

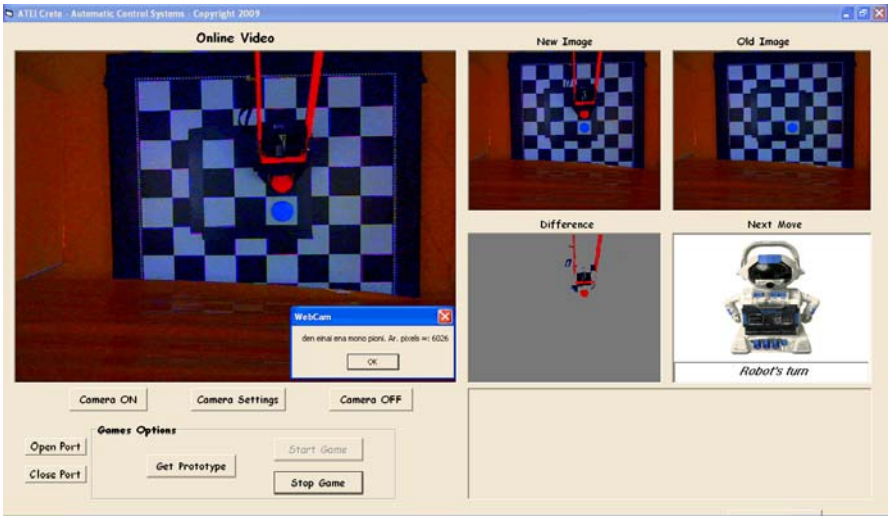
3.7 ΕΛΕΓΧΟΣ ΑΝΑΓΝΩΡΙΣΗΣ ΠΙΟΝΙΩΝ Η ΑΛΛΩΝ ΑΝΤΙΚΕΙΜΕΝΩΝ

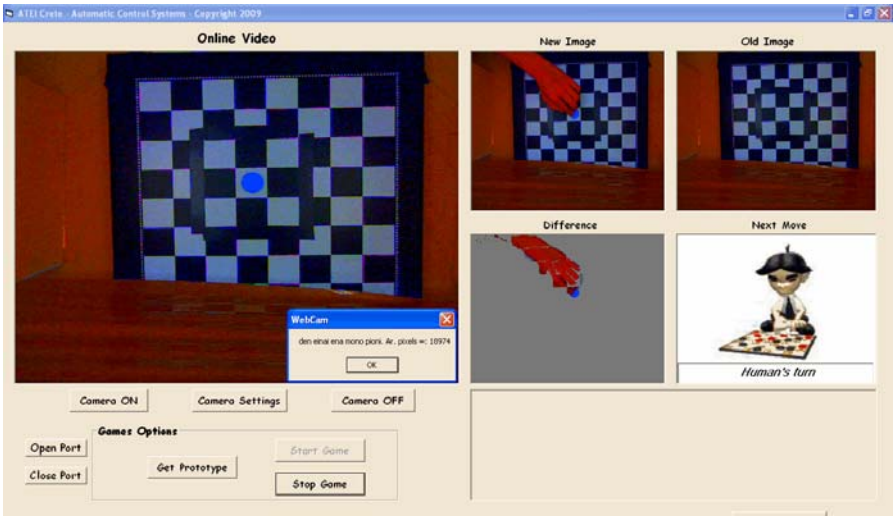
Εδώ δημιουργήθηκε το πρόγραμμα Teliko_NOP.exe το οποίο μας ενημερώνει για τον αριθμό των pixels της εικόνας result.bmp. Έτσι, το σύστημα μπορεί να γνωρίζει αν η διαφορά των εικόνων old.bmp και new.bmp είναι ίση με ένα πιόνι κάθε φορά ή υπάρχει κάτι άλλο στη φωτογραφία (όπως π.χ. το χέρι του ανθρώπου την ώρα που τοποθετεί το πιόνι ή ο ρομποτικός βραχίονας).

Για να είναι σε θέση να αντιλαμβάνεται κάτι τέτοιο, έχουμε ορίσει τα εξής

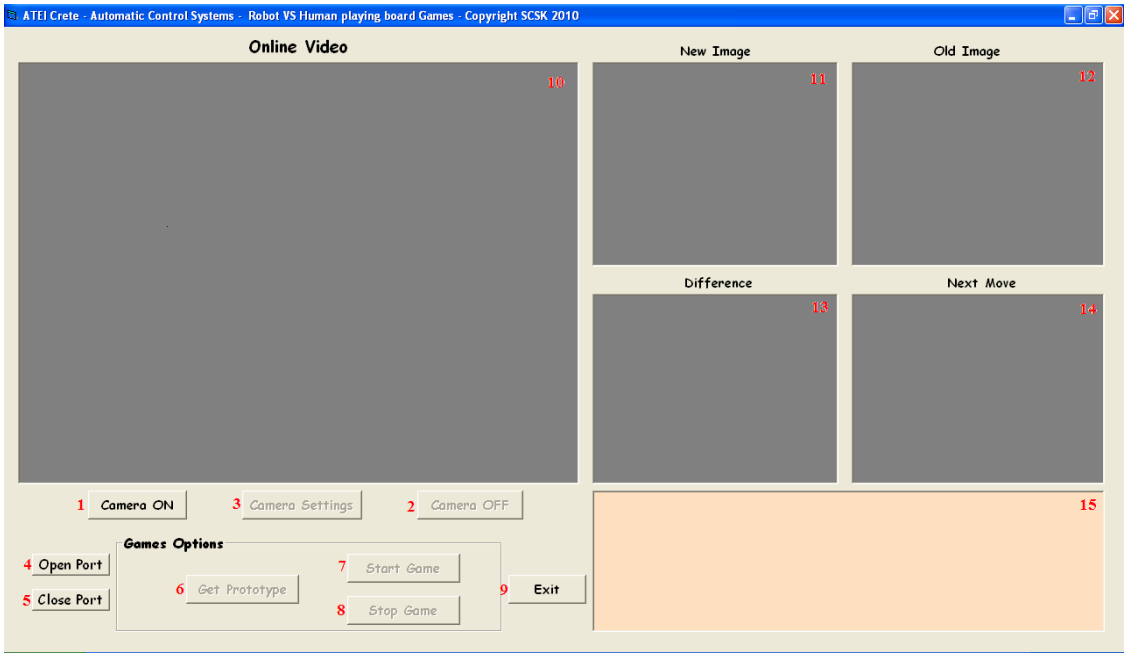
- ο αριθμος των pixels ενος πιονιου κυμαινεται απο 500 – 1000
- <500 δεν υπάρχει πιόνι
- 1000 – 1600 έχουν τοποθετηθεί δύο πιόνια
- >1600 εχει φωτογραφηθεί το χέρι ή τμήμα του βραχίονα

Έτσι δημιουργούμε το αρχείο NOP.txt στον C:\ με τον ακριβή αριθμό των pixels που βρήκε.





3.8 FRONT END ΠΕΡΙΒΑΛΛΟΝ ΓΙΑ ΤΟΝ ΕΛΕΓΧΟ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ



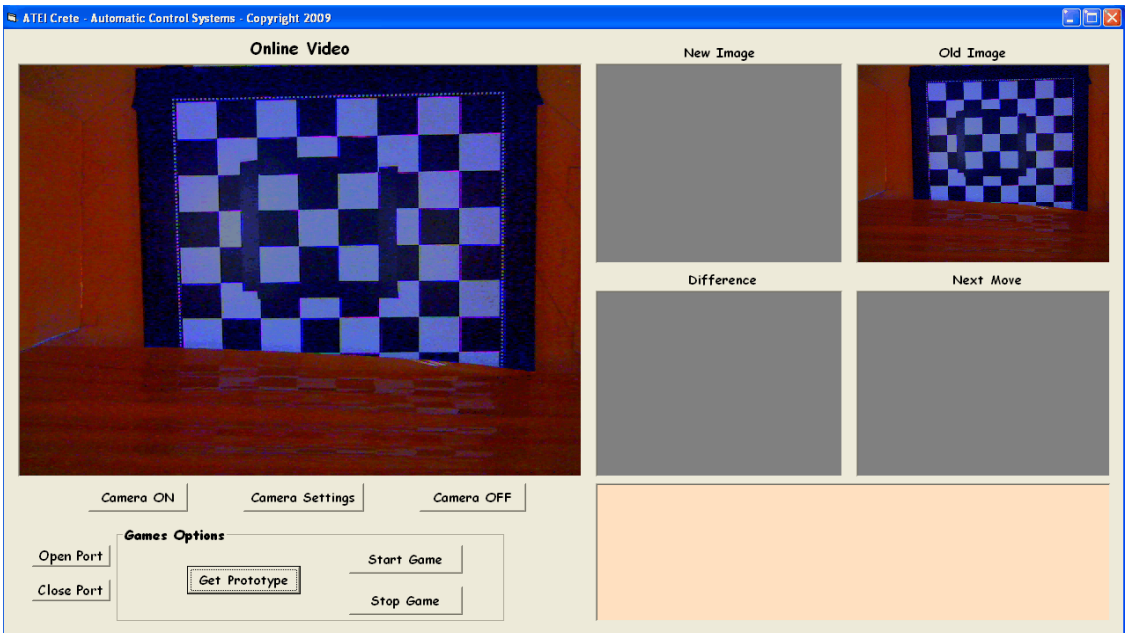
1. Ενεργοποίηση κάμερας
2. Τερματισμός κάμερας
3. Ρυθμίσεις κάμερας
4. Ενεργοποίηση Σειριακής Θύρας
5. Απενεργοποίηση Σειριακής Θύρας
6. Παίρνει φωτογραφία με την σκακιέρα κενή
7. Ξεκινάει η παρτίδα
8. Τελειώνει την παρτίδα
9. Εξοδος από την εφαρμογή
10. Εμφανίζει το Live Video
11. Εμφανίζει την νέα εικόνα
12. Εμφανίζει την παλιά εικόνα
13. Εμφανίζει την διαφορά των δύο εικόνων
14. Εμφανίζει την σειρά του παίκτη
15. Εμφανίζει μυνήματα πληροφοριών

3.9 ΔΙΑΔΙΚΑΣΙΑ AUTONOMUS VERSION

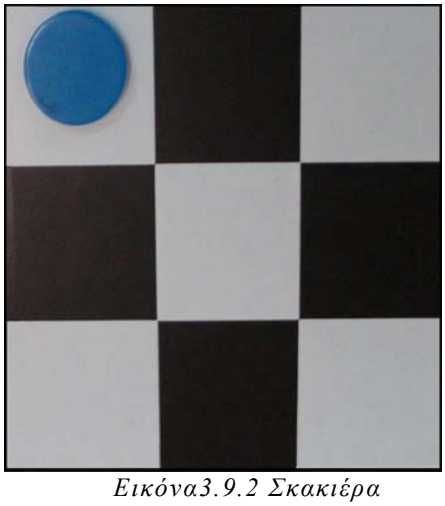
- Camera On.
- Open Port.
- Get Prototype (την αποθηκεύει με την ονομασία old.bmp).
- Start Game.
- Διαβάζει το αρχείο player.txt αν η τιμή του είναι ΄1΄ τοτε παίζει πρώτος ο άνθρωπος αν η τιμή του είναι ΄2΄ τότε παίζει πρώτος ο βραχίονας.
- Μας ενημερώνει με το αντίστοιχο μύνημα «HUMAN'S TURN» για τον πρώτο και «ROBOT'S TURN» για τον δεύτερο,εμφανίζοντας και την αντίστοιχη εικόνα.
- Τραβάει φωτογραφία και την αποθηκεύει με την ονομασία new.bmp.
- Συγκρίνει τις φωτογραφίες και δημιουργεί τις εικόνες result.bmp και result.raw.
- Διαβάζει την εικόνα result.raw.
- Καλεί το πρόγραμμα TELIKO_NOP.exe το οποίο ελέγχει αν υπάρχει μόνο ένα πιόνι (Παράγραφος 3.7).
- Καλεί το πρόγραμμα TELIKO_CENTER.exe βρίσκει το κεντρικό pixel του πιονιού αλλά και σε ποιο κουτί της σκακιέρας τοποθετήθηκε.και αποθηκευει την τιμή αυτή στο αρχείο positionh.txt .
- Καλεί το πρόγραμμα TicTacToe_CSK_190510_AUTO.exe που περιέχει τον λογικό αλγόριθμο του προγράμματος .
- Ενημερώνει το αρχείο pinakas.txt με την κίνηση που έγινε και από ποιόν έτσι ώστε το σύστημα να γνωρίζει ποιες θέσεις είναι ελεύθερες.
- Ενημερώνει το αρχείο player.txt ότι ο επόμενος παίκτης είναι ο ρομποτικός βραχίονας και εμφανίζει το μήνυμα ¨ROBOT'S TURN¨ και την εικόνα του robot.
- Καλεί ξανά το κυρίως πρόγραμμα TicTacToe_CSK_190510_AUTO.exe
- Αποφασίζει ο βραχίονας σε ποια θέση της σκακιέρας θα βάλει το πιόνι
- Ενημερώνει το αρχείο robotm.txt με την τιμή που αντιστοιχεί στην θέση αυτή.
- Διαβάζει τις αντίστοιχες συντεταγμένες .
- Επιλύεται το αντίστροφο κινηματικό πρόβλημα και δίνεται εντολή στον ρομποτικό βραχίονα μέσω της σειριακής θύρας από ποια θέση θα πάρει το πιόνι και σε ποια θέση θα το αφήσει.
- Επανέρχεται ο ρομποτικός βραχίονας στην θέση HOME.
- Η web camera τραβάει την καινούργια φωτογραφία..

- Συγκρίνει τις φωτογραφίες και δημιουργεί τις εικόνες result.bmp και result.raw
- Καλεί το πρόγραμμα TELIKO_NOP.exe. για να δει αν υπάρχει μόνο ένα πιόνι διαφορά σε σχέση με την προηγούμενη εικόνα.
- Γίνεται ο έλεγχος και ενημερώνει τα αρχεία player.txt, pinakas.txt και NOP.txt.

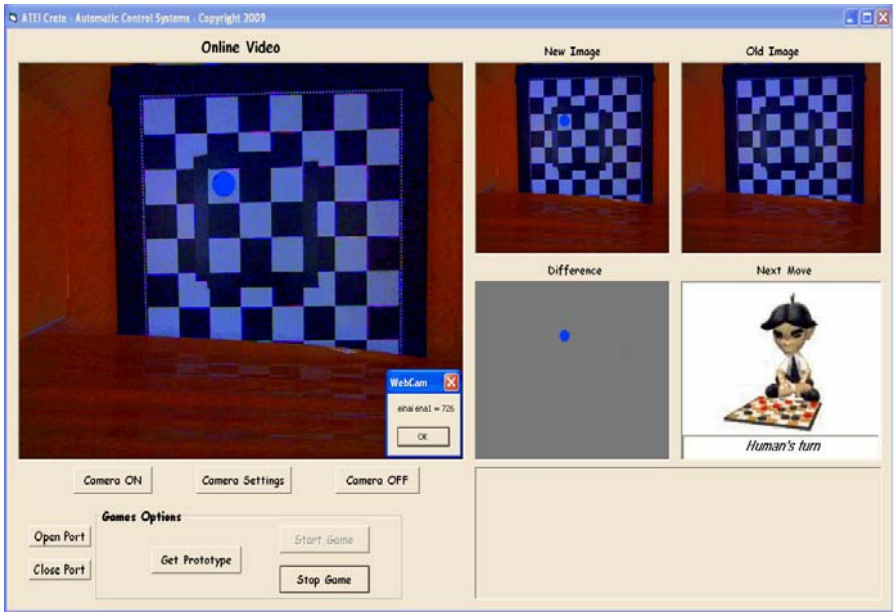
Όσο δεν υπάρχει νικητής η διαδικασία εκτελείται ξανά μεχρι να αλλάξει η τιμή του winner.txt και να μας ενημερώσει με ένα μήνυμα για το ποιος είναι ο νικητής της παρτίδας.



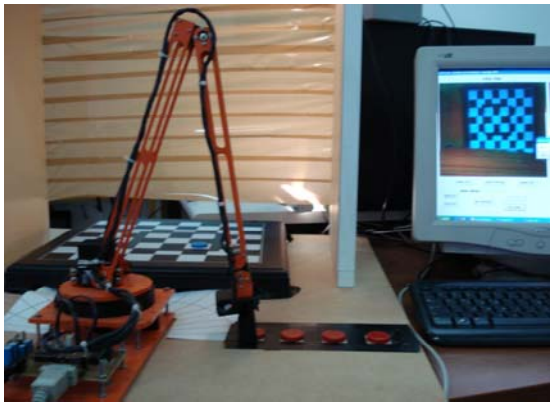
Εικόνα3.9.1 Get prototype



Εικόνα3.9.2 Σκακιέρα

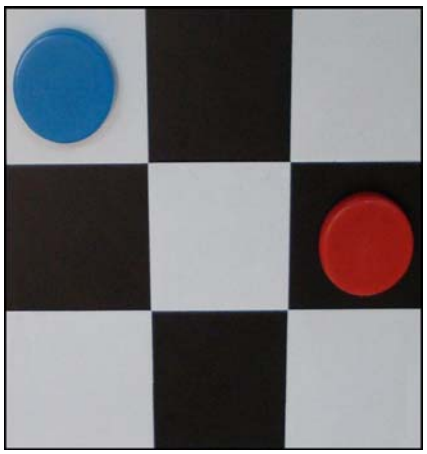


Εικόνα3.9.3 Διαφορά των δύο εικόνων στο **FRONTEND**

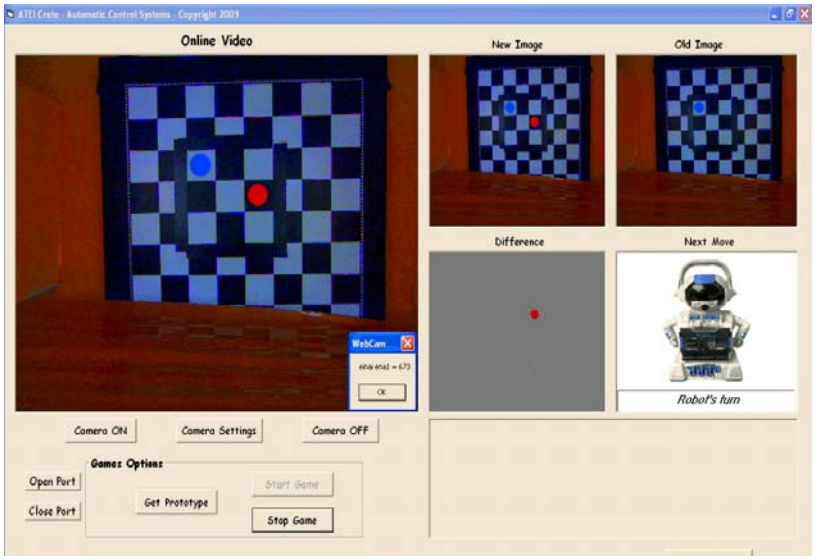


Εικόνα3.9.4 Ο βραχίονας παίρνει το πιόνι

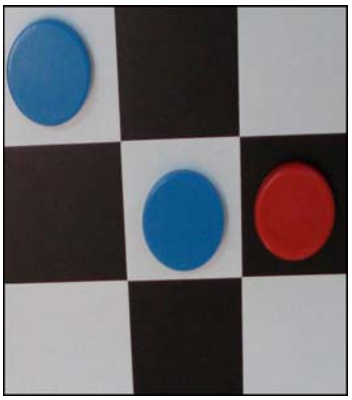
Εικόνα3.9.5 Ο βραχίονας αφήνει το πιόνι στην θέση που επέλεξε



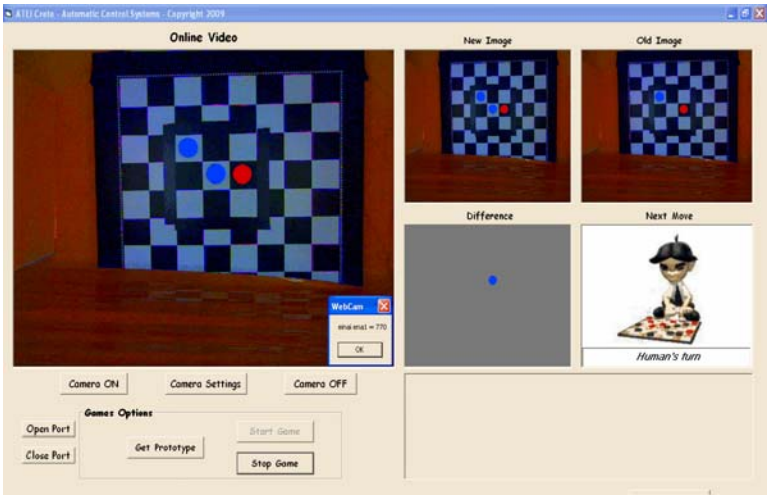
Εικόνα3.9.6 Σκακιέρα



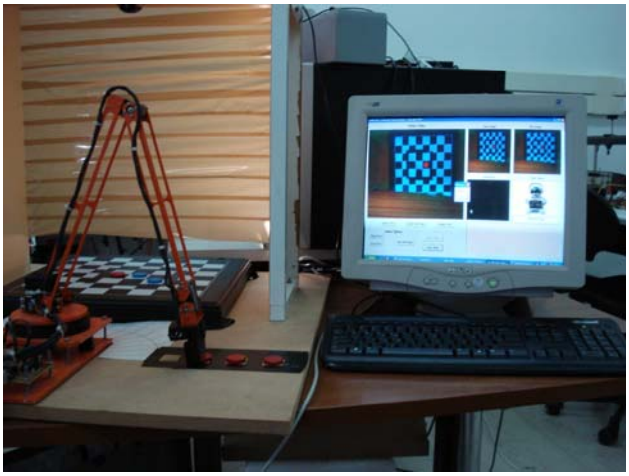
Εικόνα3.9.7 Διαφορά των δύο εικόνων στο FRONT END



Εικόνα3.9.8 Επιλογή κίνησης ανθρώπου

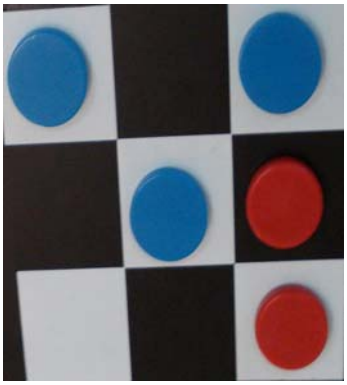


Εικόνα3.9.9 Διαφορά των δύο εικόνων στο FRONT END

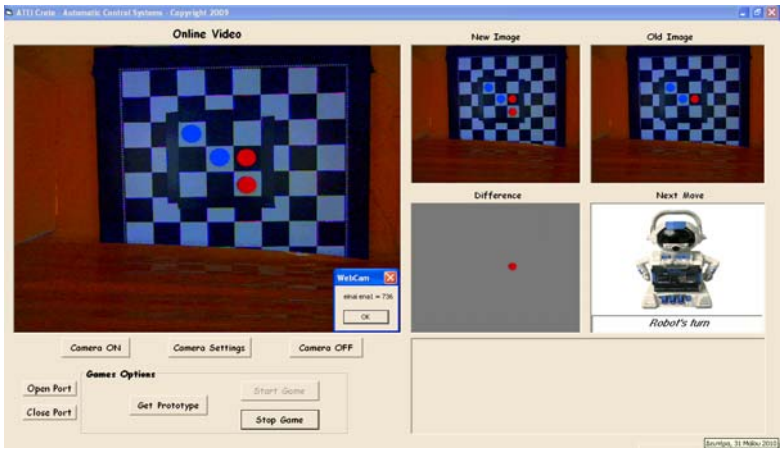


Εικόνα3.9.10 Ο βραχίονας παίρνει το πιόνι

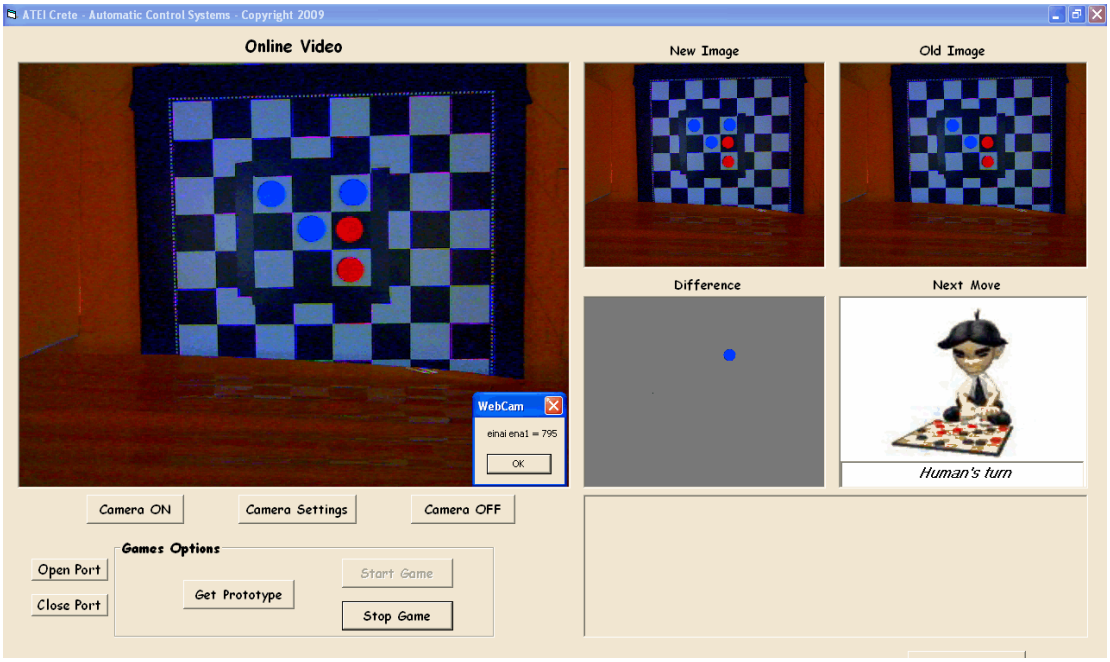
Εικόνα3.9.11 Ο βραχίονας αφήνει το πιόνι στην θέση που επέλεξε



Εικόνα3.9.12 Επιλογή κίνησης βραχίονα

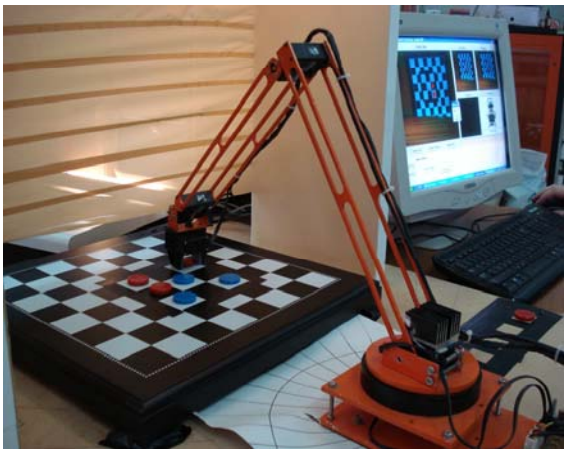


Εικόνα3.9.13 Διαφορά των δύο εικόνων στο FRONT END

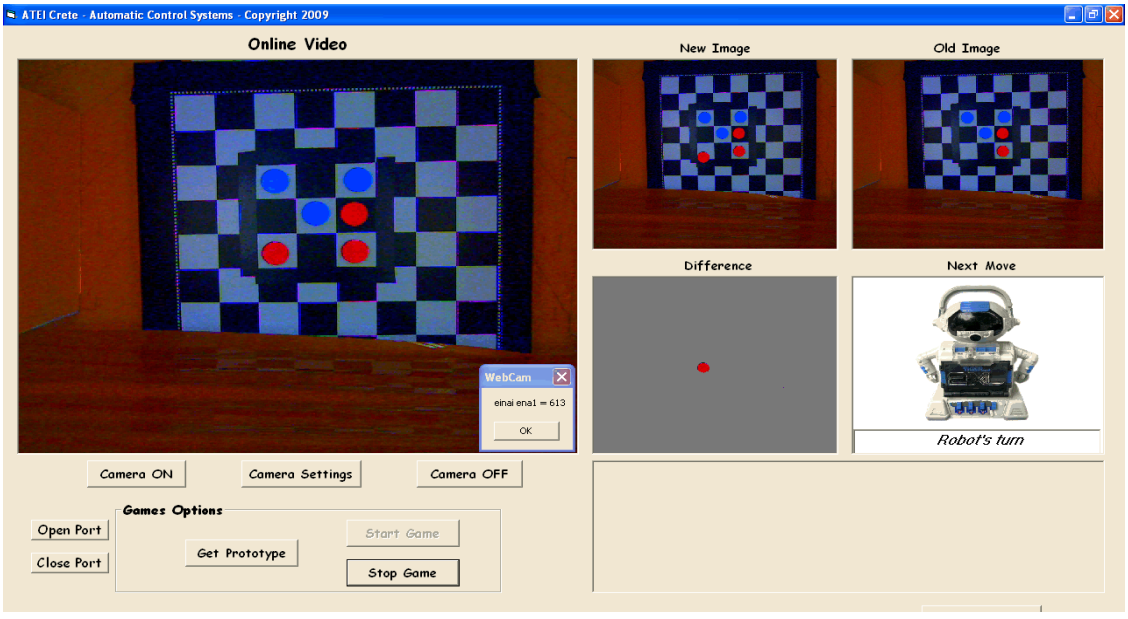


Εικόνα3.9.14 Διαφορά των δύο εικόνων στο FRONT END

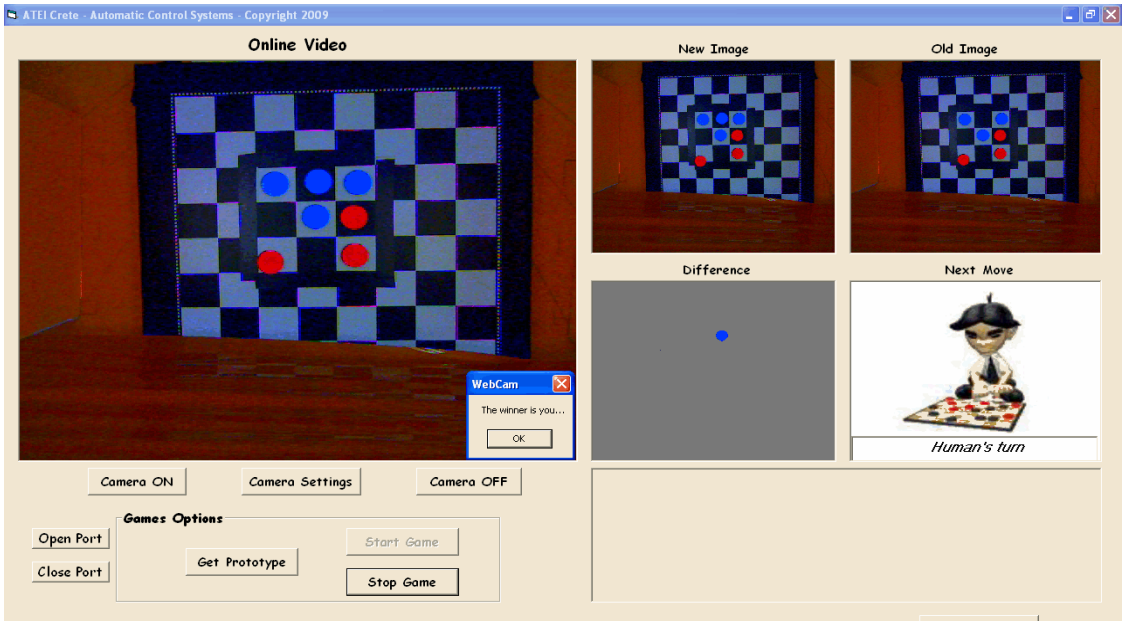
Εικόνα3.9.15 Ο βραχίονας παίρνει το πιόνι



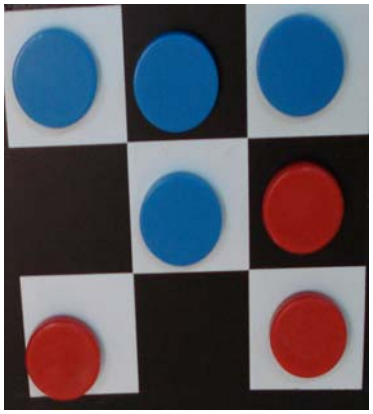
Εικόνα3.9.16 Ο βραχίονας αφήνει το πιόνι στην θέση πουεπέλεξε



Εικόνα3.9.17 Διαφορά των δύο εικόνων στο FRONT END



Εικόνα3.9.18 Διαφορά των δύο εικόνων στο FRONT END



Εικόνα3.9.19 Επιλογή κίνησης ανθρώπου

ΚΕΦΑΛΑΙΟ 4

ΠΡΟΒΛΗΜΑΤΑ - ΛΥΣΕΙΣ

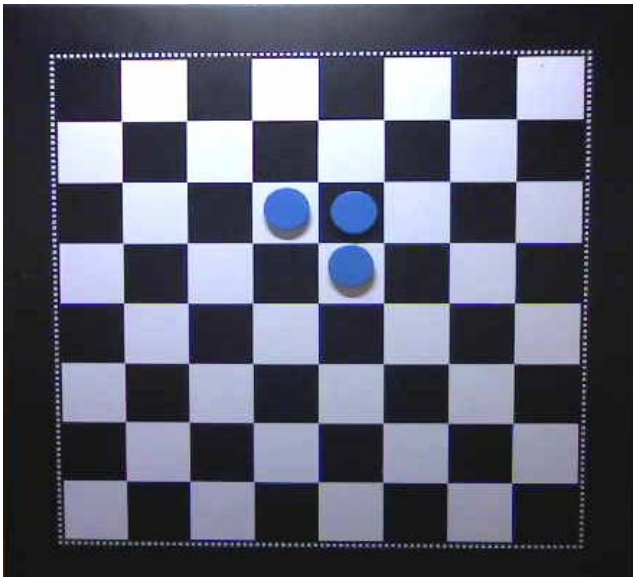
4.1 ΕΙΣΑΓΩΓΗ

Κατά τη διάρκεια εκπόνησης της πτυχιακής εργασίας παρουσιάστηκαν πολλά προβλήματα που είχαν κυρίως να κάνουν με την επεξεργασία των εικόνων. Οι λόγοι ήταν διάφοροι, ένας από τους σημαντικότερους ήταν ο κακός φωτισμός. Τον σημαντικότερο ρόλο στην επεξεργασία εικόνας τον παίζει ο φωτισμός. Η πηγή φωτός θα πρέπει να είναι τοποθετημένη έτσι ώστε να μην δημιουργούνται σκιές γύρω από τα αντικείμενα τα οποία θέλουμε να επεξεργαστούμε για να πραγματοποιηθεί αυτό θα πρέπει η πηγή φωτός να είναι τοποθετημένη έτσι ώστε το φώς να κατανέμεται ομοιόμορφα, σε όλη την εικόνα αλλά και γύρω από τα αντικείμενα προς επεξεργασία. Δεν φτάνει όμως ο φωτισμός να είναι ομοιόμορφος, πρέπει να μην επηρεάζεται και από εξωτερικούς παράγοντες, όπως το φως της ημέρας αλλά και από άλλες πηγές φωτός στον χώρο, γι αυτό λοιπόν πρέπει να είναι σταθερές οι συνθήκες φωτισμού στο σύστημα μας.

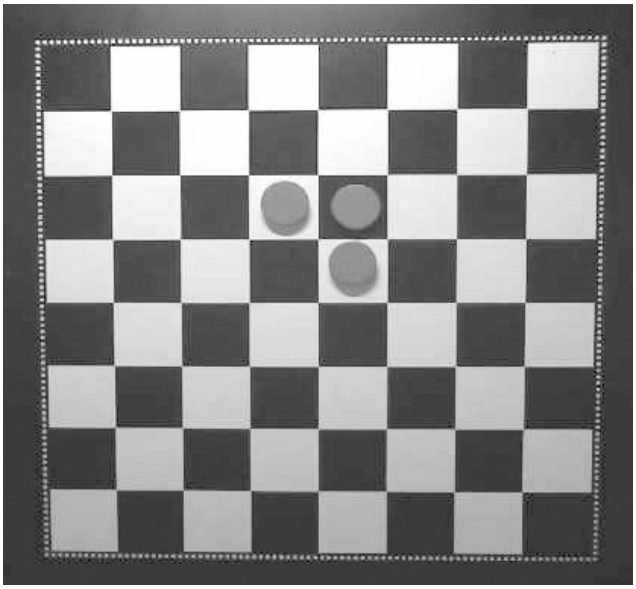
.

4.2 ΦΩΤΙΣΜΟΣ

Ένα από τα σημαντικότερα προβλήματα ήταν ο φωτισμός. Όταν τοποθετούσαμε τον φωτισμό μπροστά από την σκακιέρα δημιουργούσε σκιά, το οποίο ήταν ανεπιθύμητο γιατί σε grayscale την σκιά την αναγνώριζε σαν πιόνι με αποτέλεσμα να μην γίνεται σωστά η επεξεργασία και η αναγνώριση προτύπων.

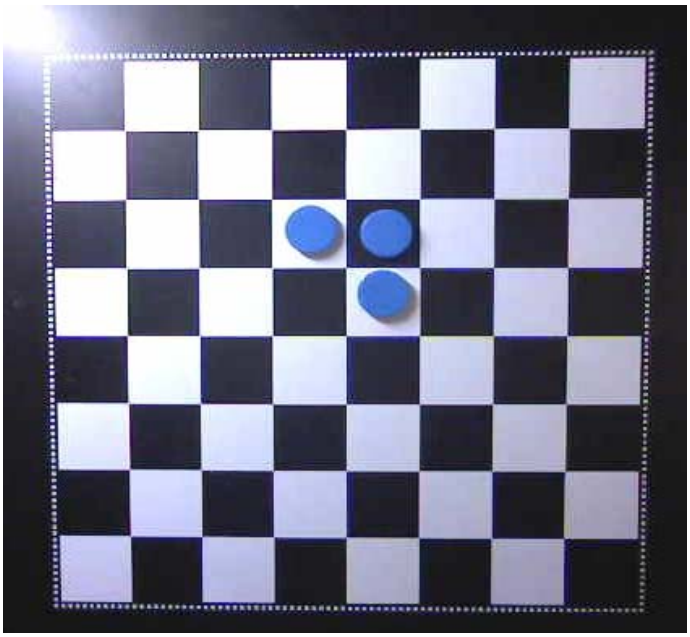


Εικόνα 4.2.1 σκιά δίπλα από τα πιόνια

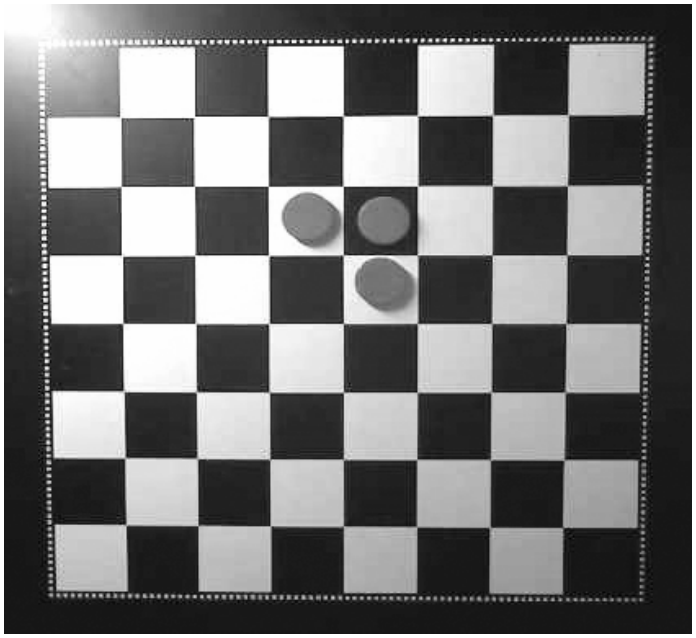


Εικόνα 4.2.2 Grayscale εικόνα με σκιά

Ο φωτισμός διαγώνια της σκακιέρας, εύκολα παρατηρείται η δημιουργία σκιάς δίπλα από τα πιόνια.

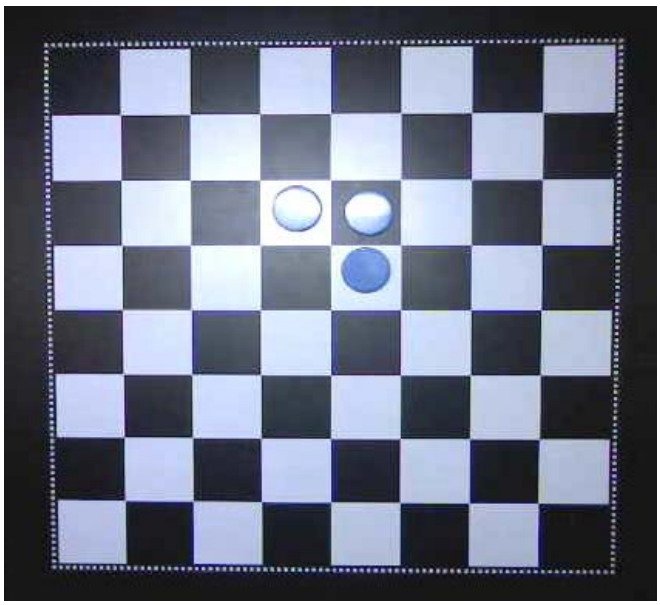


Εικόνα 4.2.3 Φωτισμός διαγώνια στην σκακιέρα και δημιουργία σκιάς

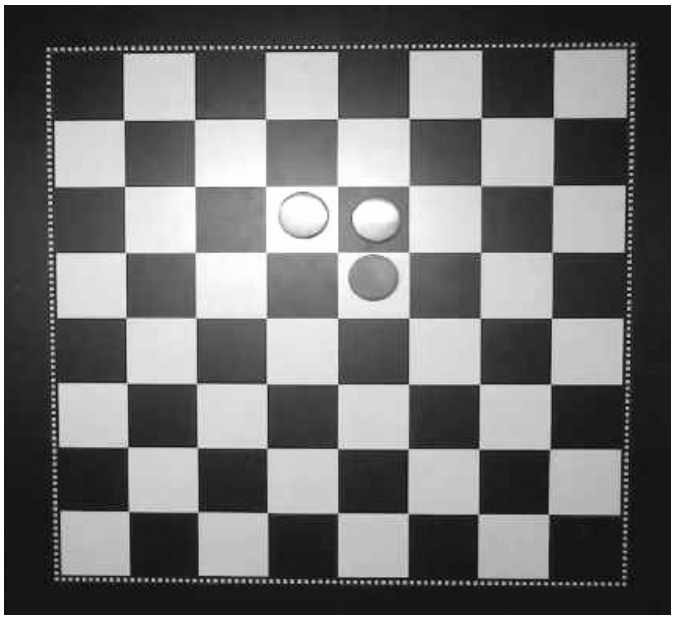


Εικόνα 4.2.4 Grayscale εικόνα με σκιά

Ο φωτισμός πάνω από την σκακιέρα δημιουργούσε επίσης σημαντικό πρόβλημα. Όπως φαίνεται και από τις παρακάτω εικόνες δημιουργούνταν αντανακλάσεις πάνω στα πιόνια αλλά και σε τμήματα της σκακιέρας με αποτέλεσμα τα gray levels να είναι πολύ κοντά στο 255(άσπρο χρώμα) έτσι η σωστή αναγνώριση των προτύπων να είναι αδύνατη.



Εικόνα 4.2.5 Αντανακλάσεις πάνω στα πιόνια



Εικόνα 4.2.6. Grayscale εικόνα με αντανακλάσεις πάνω στα πιόνια

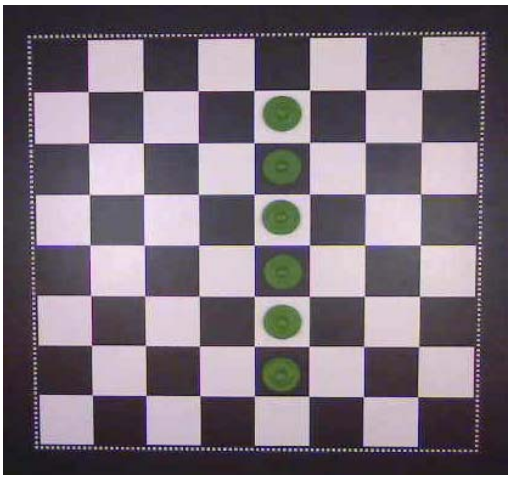
4.2.1 ΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΦΩΤΙΣΜΟΥ

Ένας άλλος σημαντικός παράγοντας ήταν οι συνθήκες φωτισμού του χώρου. Το φώς της ημέρας, με τις εναλλαγές του κατά την διάρκεια της ημέρας , άλλαζε και την συμπεριφορά του συστήματος, αφού έπρεπε κάθε φορά να αλλάζουμε τα threshold στον κώδικα μας πράγμα το οποίο ήταν ανεπιθύμητο. Για να μπορέσουμε να αποφύγουμε όλα τα παραπάνω προβλήματα του φωτισμού, στο πάνω μέρος του πλαισίου τοποθετήσαμε ένα πανί ικανό να απορροφήσει το πολύ φώς, ώστε να μην δημιουργούνται αντανακλάσεις του φωτός πάνω στα πιόνια ή στην σκακιέρα. Και πάνω από το πανί τοποθετήσαμε τις λάμπες φθορισμού, ώστε το φως να πέφτει ομοιόμορφα στην σκακιέρα και να μην δημιουργούνται σκιές γύρω από τα πιόνια (soft light). Για να περιορίσουμε τον ανεπιθύμητο φωτισμό από τους εξωτερικούς παράγοντες, καλύφθηκαν τα πλαϊνά και το πίσω μέρος του πλαισίου στήριξης της κάμερας. Ένα άλλο εξίσου σημαντικό πρόβλημα ήταν τα gray levels που ήταν πολύ κοντά το ένα με το άλλο.

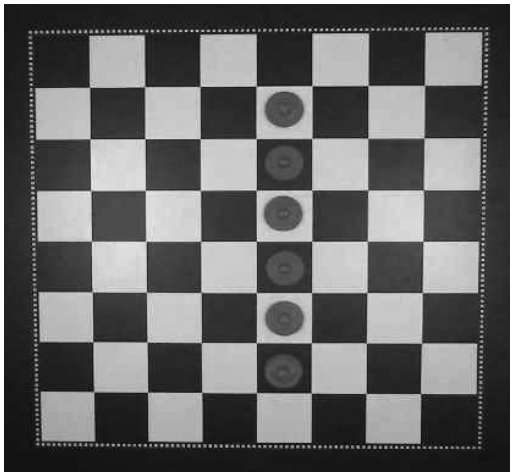
4.3 GRAYLEVELS

Σημαντικό πρόβλημα μας δημιούργησαν τα gray levels των χρωμάτων, που ήταν αρκετά κοντά το ένα με το άλλο με αποτέλεσμα όταν τοποθετούσαμε το κατώφλι να μας εμφανίζονται και οι δύο αποχρώσεις των πιονιών αλλά και μέρος των κουτιών της σκακιέρας.

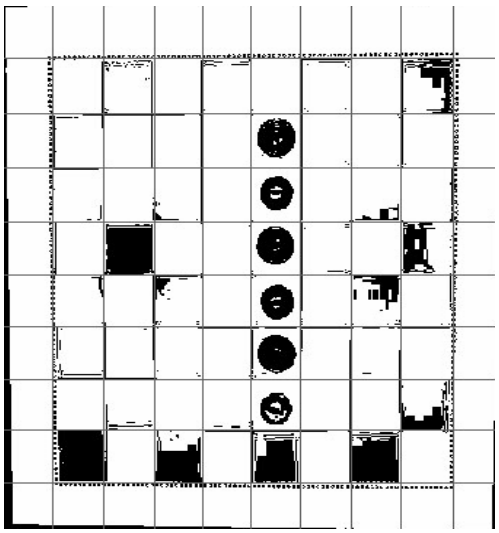
Για τα πράσινα πιόνια εύκολα παρατηρεί κάνεις ότι το επίπεδο του γκρι που αντιστοιχεί στο πράσινο χρώμα είναι πολύ κοντά στο επίπεδο του γκρι που αντιστοιχεί στο μαύρο χρώμα του κουτιού της σκακιέρας αποτέλεσμα όταν από την εικόνα περάσουμε τον αλγόριθμο που δημιουργήσαμε και τοποθετήσουμε το κατώφλι, να παίρνουμε την εικόνα (4.3.3).



Εικόνα 4.3.1



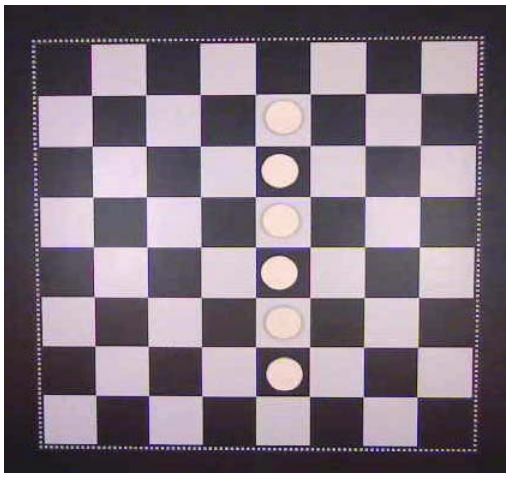
Εικόνα 4.3.2



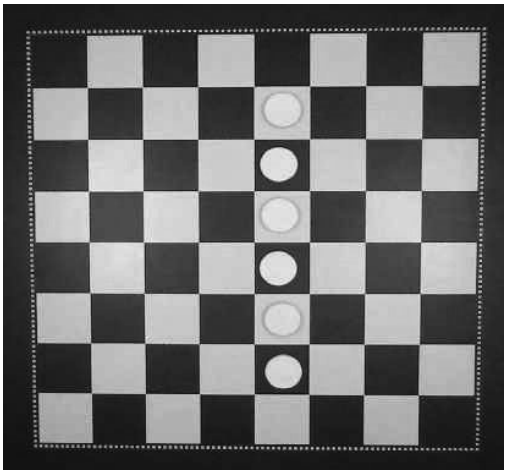
Εικόνα 4.3.3

Το κατώφλι στην εικόνα 4.3.3 έχει οριστεί από 70-180, δηλαδή να εμφανίζει στην καινούργια εικόνα τις τιμές του γκρι από 70 έως 180, περίπου τις τιμές που αντιστοιχούν στο πράσινο χρώμα. Στην πραγματικότητα όμως δεν μας εμφάνιζε μόνο την γκρι απόχρωση του πράσινου χρώματος. Τα λευκά κενά που υπάρχουν πάνω στα πιόνια έχουν δημιουργηθεί λόγων των αντανακλάσεων του φωτισμού πάνω στο πιόνι.

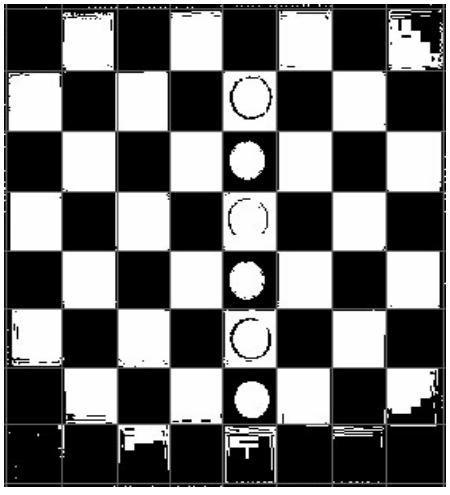
Στα λευκά πιόνια το επίπεδο του γκρι είναι σχεδόν το ίδιο με αυτό του λευκού χρώματος των κουτιών της σκακιέρας με αποτέλεσμα το πιόνι μετά την επεξεργασία και την τοποθέτηση του κατωφλίου στο διάστημα 180 έως 240 να μην φαίνεται καθόλου όπως φαίνεται στην εικόνα (4.3.6)



Εικόνα 4.3.4

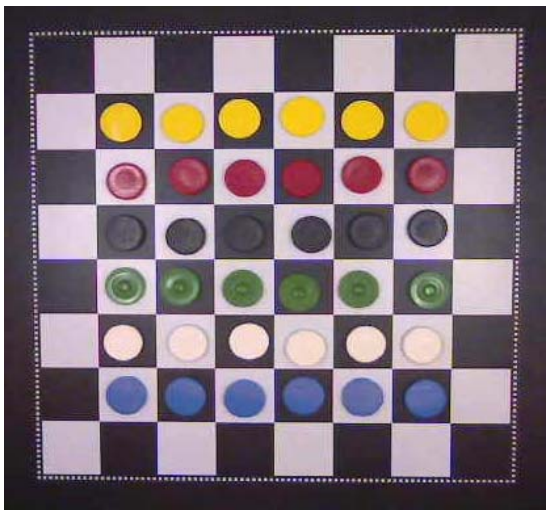


Εικόνα 4.3.5

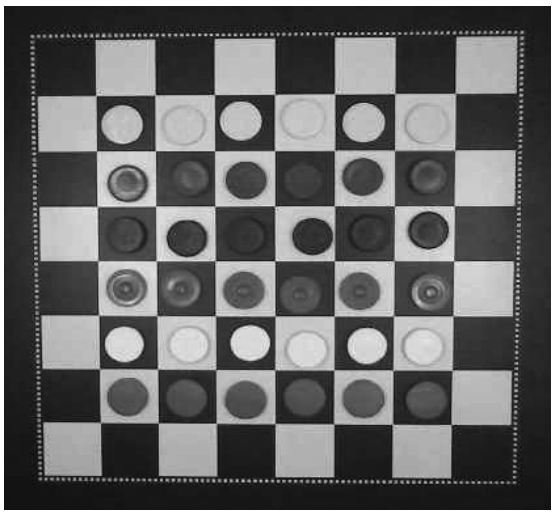


Εικόνα 4.3.6

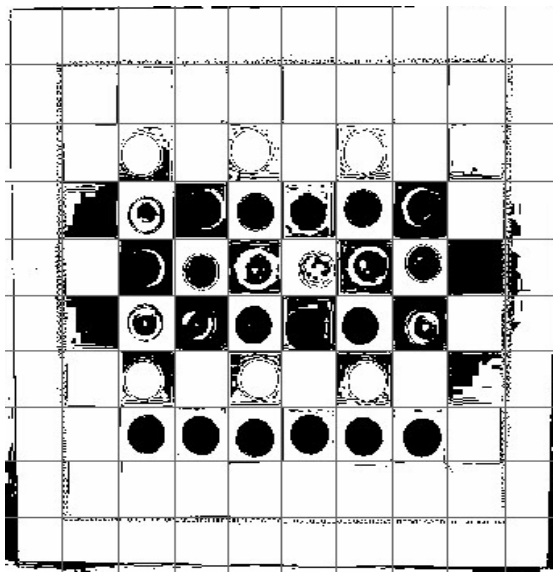
Στην παρακάτω εικόνα έχουμε συγκεντρώσει όλες τις χρωματικές αποχρώσεις των πιονιών με κατώφλι 50 έως 110



Εικόνα 4.3.7



Εικόνα 4.3.8



Εικόνα 4.3.9

4.3.1 ΛΥΣΗ ΠΡΟΒΛΗΜΑΤΟΣ ΜΕ ΤΑ GRAY LEVELS

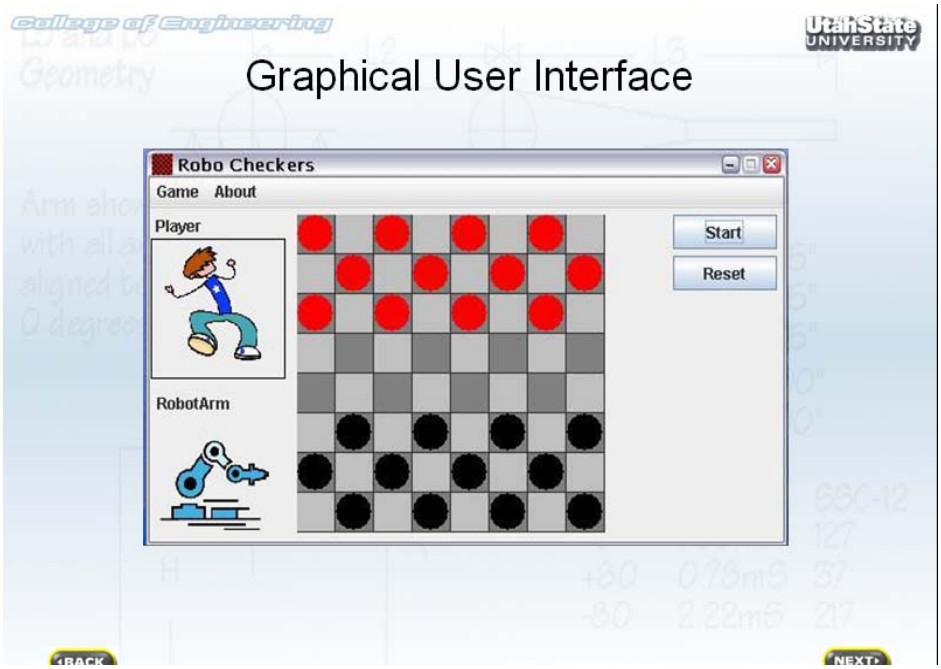
Για να λύσουμε το πρόβλημα αυτό επιλέξαμε χρώματα όπου οι αποχρώσεις του γκρι τους δεν είναι κοντά ούτε στο άσπρο αλλά ούτε και στο μαύρο. Τέτοια χρώματα είναι το μπλε και το κόκκινο που με το κατάλληλο κατώφλι και φωτισμό (soft light) δεν μας δημιουργούσαν πρόβλημα στην επεξεργασία της εικόνας.

ΚΕΦΑΛΑΙΟ 5

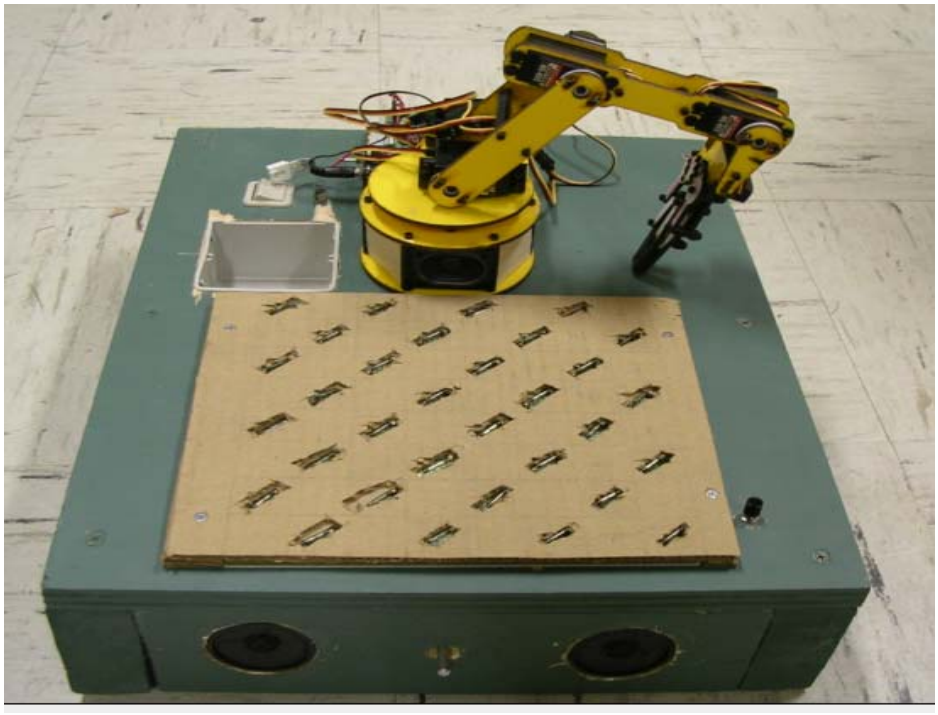
ΠΑΡΟΜΟΙΑ ΡΟΜΠΟΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

5.1 ΠΑΝΕΠΙΣΤΗΜΙΟ UTAH

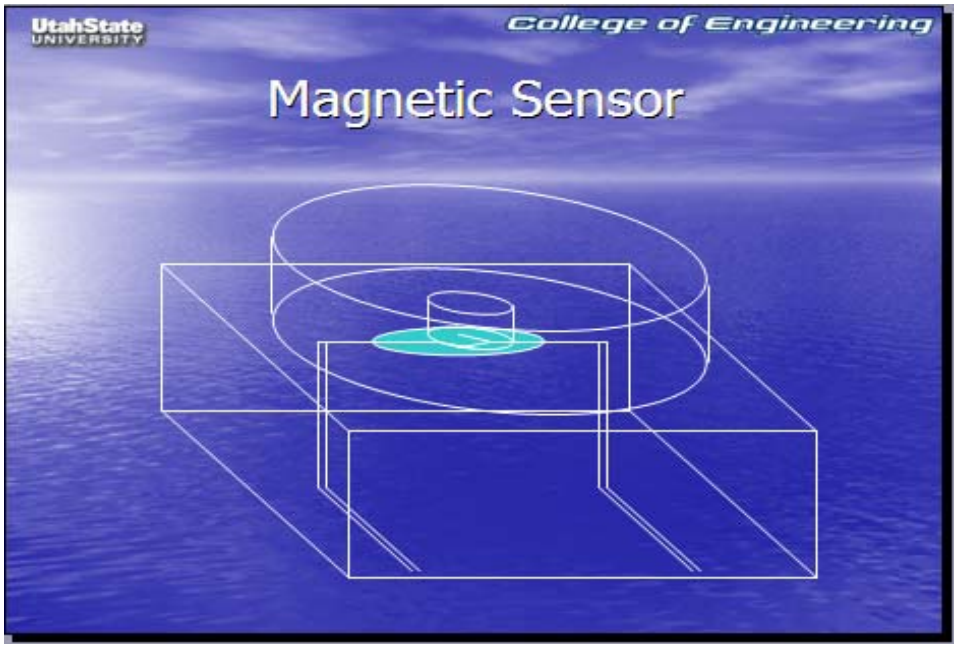
Μία παρόμοια προσπάθεια έγινε στο πανεπιστήμιο UTAH, όπου κατασκευάστηκε ένα σύστημα το οποίο όμως δεν είναι αυτόνομο. Ο πρώτος παίχτης είναι ο άνθρωπος, όταν τοποθετήσει ο πρώτος παίχτης το πιόνι πάνω στην σκακιέρα τότε η κίνηση του αναγνωρίζεται από έναν αισθητήρα που βρίσκεται σε κάθε κουτί της σκακιέρας. Ο δεύτερος παίχτης είναι θεωρητικά το robot που κατασκευάστηκε, πρακτικά όμως το robot δεν έχει την δυνατότητα λήψης αποφάσεων γι' αυτό και κινείται με την εντολή του δεύτερου παίχτη, η οποία δίδεται χειροκίνητα από το εικονικό περιβάλλον που δημιουργήθηκε.



Εικόνα 5.1.1 Γραφικό περιβάλλον



Εικόνα 5.1.1 Μαγνητικές επαφές συστήματος



Εικόνα 5.1.1 Διάταξη μαγνητικών επαφών

5.2 STAIR ROBOTIC SYSTEM

Το ρομποτικό σύστημα Stanford Artificial Intelligence Robot (STAIR) κατασκευάστηκε από το Stanford University, ένα project που ξεκίνησε ο καθηγητής Andrew Ng και η ομάδα του.

Το συγκεκριμένο ρομποτικό σύστημα έχει σχεδιαστεί να βοηθάει στις δουλειές του σπιτιού. Έχοντας τοποθετημένη μία camera μπορεί να βλέπει ,κινείται στον χώρο με την βοήθεια δύο τροχών και μπορεί να πιάνει και να μεταφέρει αντικείμενα χρησιμοποιώντας τον ρομποτικό βραχίονα που έχει τοποθετηθεί κοντά στην βάση του συστήματος.

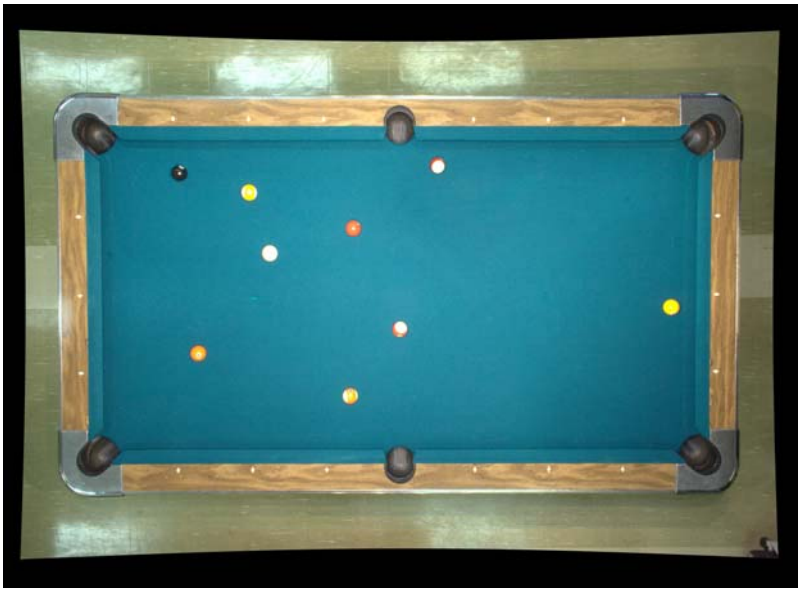
Είναι ικανό να φέρνει ή να μετακινεί αντικείμενα που του ζητούνται στον χώρο. Καθώς έχει μία βάση με φωτογραφίες διάφορων αντικειμένων ώστε να είναι σε θέση να αναγνωρίζει το αντικείμενο και να υπολογίζει τον πιο σωστό τρόπο να το πιάσει.

Εικόνα 5.2.1 STAIR Robotic system

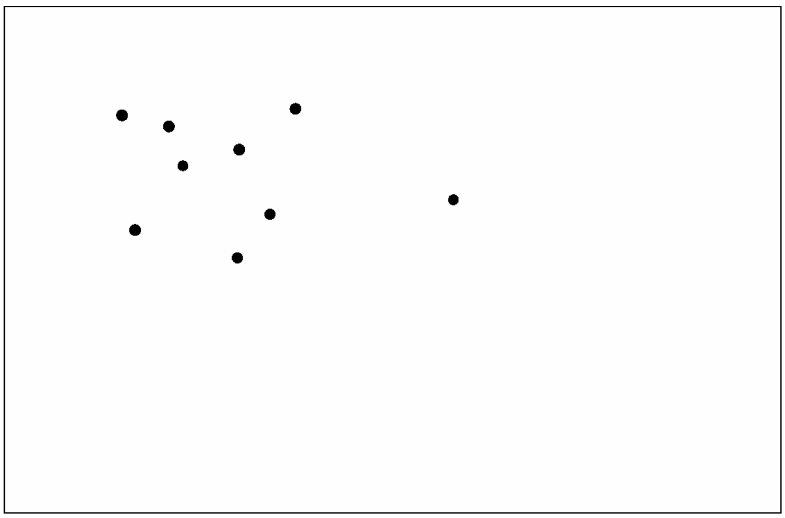
5.3 ΡΟΜΠΟΤΙΚΟ ΣΥΣΤΗΜΑ ΠΑΙΖΕΙ ΜΠΙΛΙΑΡΔΟ

Το σύστημα που δημιουργήθηκε από το QUEEN'S UNIVERSITY, CANADA είναι εξυπνο σύστημα το οποίο παίζει μπιλιάρδο με αντίπαλο τον άνθρωπο, αναγνωρίζει τις μπάλες πάνω στο τραπέζι και εντοπίζει την ακριβή τους θέση χρησιμοποιώντας δύο κάμερες.

Εικόνα 5.3.2.1

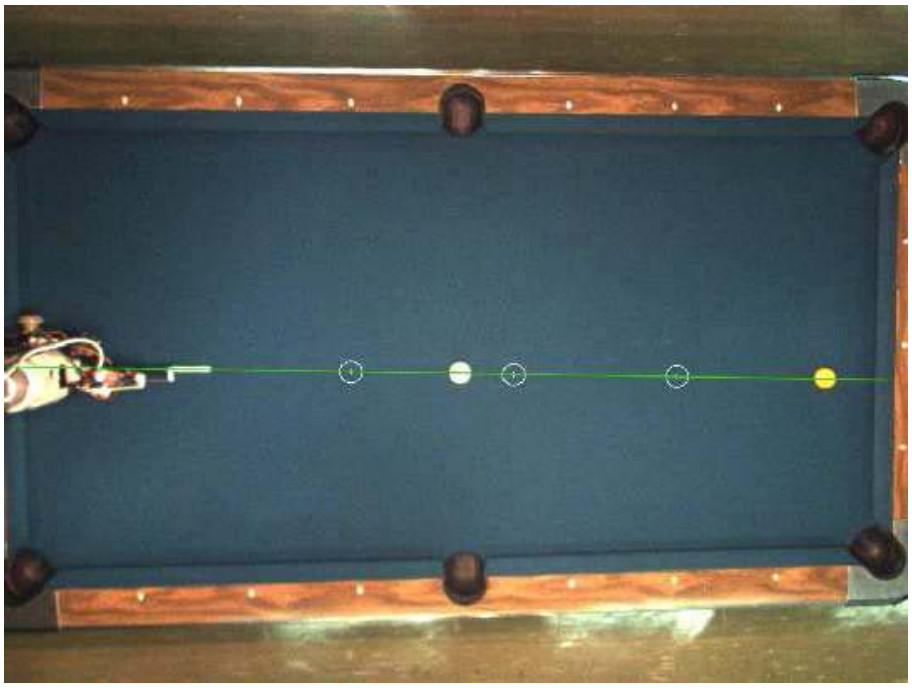


Εικόνα 5.3.2.1 Εικόνα του συστήματος raw



Εικόνα 5.3.2.2 Επεξεργασμένη εικόνα

Εικόνα 5.3.2.3 Ρομποτικό σύστημα



Εικόνα 5.3.2.4Εντοπισμός βέλτιστου χτυπήματος

ΒΙΒΛΙΟΓΡΑΦΙΑ

http://www.wikipedia.org

http://www.stair.stanford.edu/index.php

http://www.medgreece.gr

Ψηφιακή Επεξεργασία Εικόνας, Ιωάννης Πήτας , 2001

Digital Image Processing Third Edition, R.C. Gonzalez and R.E. Woods, 2008

Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας,Παπαμακάριος Νικόλαος, 2005

Σημειώσεις Ψηφιακής Επεξεργασίας Εικόνας και Ήχου, Εμμανουήλ Δ.Σκουνάκης, 2008

ΠΑΡΑΡΤΗΜΑ

ΚΩΔΙΚΑΣ

Παρακάτω παρατίθεται ο κώδικας του συστήματος:

TICTACTOE_CSK_190510_AUTO.EXE :

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{ int i = 0, j = 0, y = 0, z = 0 ;
 // Loop counters int player = 0, met = 0 ;
 // Player number- 1(human-manual) or 2(robot-automatic)  int go = 0;
 // Square selection number int winner = 0;
 // The winning player int choice, kinisi; int pinakas[3][3]; char board[3][3] = {
 // The board
                  {'1','2','3'},        // Initial values are reference numbers
                  {'4','5','6'},        // used to select a vacant square for
                  {'7','8','9'}         // a turn.
                 };
 FILE *fp1,*fp2,*fp3,*fp4,*fp5,*fp6,*fp7,*fp8;

 kinisi=0;
 fp1=fopen("c:\\player.txt","rb");  fscanf(fp1,"%d",&player);  fclose(fp1);
 for (i=0;i<3;i++){
     for (j=0;j<3;j++){
         pinakas[i][j]=0;
```

```
        }}

 fp2=fopen("c:\\pinakas.txt","rb");

     for (j=0;j<3;j++) {fscanf(fp2,"%d",&pinakas[i][j]);if (pinakas[0][0]>2)
pinakas[0][0]=0;
        if (pinakas[i][j]!=0) board[i][j]='-';} }
 fclose(fp2);

    choice=0;

    /* Get valid player square selection */
   do
    {
     if (player==1)
     {
      fp6=fopen("c:\\positionh.txt","rb"); fscanf(fp6,"%d",&go);
        fclose(fp6);} else {
        fp5=fopen("c:\\kinisi.txt","rb");
          fscanf(fp5,"%d",&kinisi);
        fclose(fp5);

       fp5=fopen("c:\\kinisi.txt","wb"); fprintf(fp5,"%d",kinisi+1); fclose(fp5);

      if (pinakas[0][0]==2 && pinakas[0][1]==2 && pinakas[0][2]==0){ go=3; z=1;}
      if (pinakas[0][0]==2 && pinakas[0][2]==2 && pinakas[0][1]==0){ go=2; z=1;}
      if (pinakas[0][1]==2 && pinakas[0][2]==2 && pinakas[0][0]==0){ go=1; z=1;}
      if (pinakas[1][0]==2 && pinakas[1][1]==2 && pinakas[1][2]==0){ go=6; z=1;}
      if (pinakas[1][0]==2 && pinakas[1][2]==2 && pinakas[1][1]==0){ go=5; z=1;}
      if (pinakas[1][1]==2 && pinakas[1][2]==2 && pinakas[1][0]==0){ go=4; z=1;}
      if (pinakas[2][0]==2 && pinakas[2][1]==2 && pinakas[2][2]==0){ go=9; z=1;}
      if (pinakas[2][0]==2 && pinakas[2][2]==2 && pinakas[2][1]==0){ go=8; z=1;}
```

```
if (pinakas[2][1]==2 && pinakas[2][2]==2 && pinakas[2][0]==0){ go=7; z=1;}
if (pinakas[0][0]==2 && pinakas[1][1]==2 && pinakas[2][2]==0){ go=9; z=1;}
if (pinakas[0][0]==2 && pinakas[2][2]==2 && pinakas[1][1]==0){ go=5; z=1;}
if (pinakas[1][1]==2 && pinakas[2][2]==2 && pinakas[0][0]==0){ go=1; z=1;}
if (pinakas[2][0]==2 && pinakas[1][1]==2 && pinakas[0][2]==0){ go=3; z=1;}
if (pinakas[2][0]==2 && pinakas[0][2]==2 && pinakas[1][1]==0){ go=5; z=1;}
if (pinakas[1][1]==2 && pinakas[0][2]==2 && pinakas[2][0]==0){ go=7; z=1;}
if (pinakas[0][0]==2 && pinakas[1][0]==2 && pinakas[2][0]==0){ go=7; z=1;}
if (pinakas[0][0]==2 && pinakas[2][0]==2 && pinakas[1][0]==0){ go=4; z=1;}
if (pinakas[1][0]==2 && pinakas[2][0]==2 && pinakas[0][0]==0){ go=1; z=1;}
if (pinakas[0][1]==2 && pinakas[1][1]==2 && pinakas[2][1]==0){ go=8; z=1;}
if (pinakas[0][1]==2 && pinakas[2][1]==2 && pinakas[1][1]==0){ go=5; z=1;}
if (pinakas[1][1]==2 && pinakas[2][1]==2 && pinakas[0][1]==0){ go=2; z=1;}
if (pinakas[0][2]==2 && pinakas[1][2]==2 && pinakas[2][2]==0){ go=9; z=1;}
if (pinakas[0][2]==2 && pinakas[2][2]==2 && pinakas[1][2]==0){ go=6; z=1;}
if (pinakas[1][2]==2 && pinakas[2][2]==2 && pinakas[0][2]==0){ go=3; z=1;}

if (z==0 && pinakas[0][0]==1 && pinakas[0][1]==1 && pinakas[0][2]==0){ go=3; y=1;}
if (z==0 && pinakas[0][0]==1 && pinakas[0][2]==1 && pinakas[0][1]==0){ go=2; y=1;}
if (z==0 && pinakas[0][1]==1 && pinakas[0][2]==1 && pinakas[0][0]==0){ go=1; y=1;}
if (z==0 && pinakas[1][0]==1 && pinakas[1][1]==1 && pinakas[1][2]==0){ go=6; y=1;}
if (z==0 && pinakas[1][0]==1 && pinakas[1][2]==1 && pinakas[1][1]==0){ go=5; y=1;}
if (z==0 && pinakas[1][1]==1 && pinakas[1][2]==1 && pinakas[1][0]==0){ go=4; y=1;}
if (z==0 && pinakas[2][0]==1 && pinakas[2][1]==1 && pinakas[2][2]==0){ go=9; y=1;}
if (z==0 && pinakas[2][0]==1 && pinakas[2][2]==1 && pinakas[2][1]==0){ go=8; y=1;}
if (z==0 && pinakas[2][1]==1 && pinakas[2][2]==1 && pinakas[2][0]==0){ go=7; y=1;}
if (z==0 && pinakas[0][0]==1 && pinakas[1][1]==1 && pinakas[2][2]==0){ go=9; y=1;}
if (z==0 && pinakas[0][0]==1 && pinakas[2][2]==1 && pinakas[1][1]==0){ go=5; y=1;}
if (z==0 && pinakas[1][1]==1 && pinakas[2][2]==1 && pinakas[0][0]==0){ go=1; y=1;}
if (z==0 && pinakas[2][0]==1 && pinakas[1][1]==1 && pinakas[0][2]==0){ go=3; y=1;}
if (z==0 && pinakas[2][0]==1 && pinakas[0][2]==1 && pinakas[1][1]==0){ go=5; y=1;}
```

```
if (z==0 && pinakas[1][1]==1 && pinakas[0][2]==1 && pinakas[2][0]==0){ go=7; y=1;}
if (z==0 && pinakas[0][0]==1 && pinakas[1][0]==1 && pinakas[2][0]==0){ go=7; y=1;}
if (z==0 && pinakas[0][0]==1 && pinakas[2][0]==1 && pinakas[1][0]==0){ go=4; y=1;}
if (z==0 && pinakas[1][0]==1 && pinakas[2][0]==1 && pinakas[0][0]==0){ go=1; y=1;}
if (z==0 && pinakas[0][1]==1 && pinakas[1][1]==1 && pinakas[2][1]==0){ go=8; y=1;}
if (z==0 && pinakas[0][1]==1 && pinakas[2][1]==1 && pinakas[1][1]==0){ go=5; y=1;}
if (z==0 && pinakas[1][1]==1 && pinakas[2][1]==1 && pinakas[0][1]==0){ go=2; y=1;}
if (z==0 && pinakas[0][2]==1 && pinakas[1][2]==1 && pinakas[2][2]==0){ go=9; y=1;}
if (z==0 && pinakas[0][2]==1 && pinakas[2][2]==1 && pinakas[1][2]==0){ go=6; y=1;}
if (z==0 && pinakas[1][2]==1 && pinakas[2][2]==1 && pinakas[0][2]==0){ go=3; y=1;}
if(y==0 && z==0){go=rand()%9+1;}
 fp4=fopen("c:\\robotm.txt","wb");fprintf(fp4,"%d",go); fclose(fp4);
    if ((kinisi+1)==1)
    { fp7=fopen("c:\\pare.txt","wb");
      fprintf(fp7,"%f\n %f\n %f",0.1380,0.0100,0.0130);
      fclose(fp7);}

    if ((kinisi+1)==2)
    { fp7=fopen("c:\\pare.txt","wb");
      fprintf(fp7,"%f\n %f\n %f",0.1850,0.0100,0.0080);
      fclose(fp7);}

    if ((kinisi+1)==3)
    { fp7=fopen("c:\\pare.txt","wb");
      fprintf(fp7,"%f\n %f\n %f",0.2294,0.0100,0.0027);
      fclose(fp7);}

    if ((kinisi+1)==4)
    { fp7=fopen("c:\\pare.txt","wb");
      fprintf(fp7,"%f\n %f\n %f",0.2867,0.0100,0.0000);
      fclose(fp7);}
```

```
  fp8=fopen("c:\\afise.txt","wb");
  if (go==1) fprintf(fp8,"%f\n %f\n %f",0.0420,0.3629,0.0427);
  if (go==2) fprintf(fp8,"%f\n %f\n %f",0.0100,0.3675,0.0393);
  if (go==3) fprintf(fp8,"%f\n %f\n %f",0.0652,0.3698,0.0362);
  if (go==4) fprintf(fp8,"%f\n %f\n %f",-0.0389,0.3172,0.0381);
  if (go==5) fprintf(fp8,"%f\n %f\n %f",0.0240,0.3247,0.0407);
  if (go==6) fprintf(fp8,"%f\n %f\n %f",0.0640,0.3256,0.0375);
  if (go==7) fprintf(fp8,"%f\n %f\n %f",-0.0345,0.2807,0.0423);
  if (go==8) fprintf(fp8,"%f\n %f\n %f",0.0246,0.2817,0.0423);
  if (go==9) fprintf(fp8,"%f\n %f\n %f",0.0599,0.2818,0.0446);
  fclose(fp8);
 }

if (player==1){
if ((go==1) && (pinakas[0][0]==0)) {pinakas[0][0]=1;choice=1;board[0][0]='X';}
if ((go==2) && (pinakas[0][1]==0)) {pinakas[0][1]=1;choice=1;board[0][1]='X';}
if ((go==3) && (pinakas[0][2]==0)) {pinakas[0][2]=1;choice=1;board[0][2]='X';}
if ((go==4) && (pinakas[1][0]==0)) {pinakas[1][0]=1;choice=1;board[1][0]='X';}
if ((go==5) && (pinakas[1][1]==0)) {pinakas[1][1]=1;choice=1;board[1][1]='X';}
if ((go==6) && (pinakas[1][2]==0)) {pinakas[1][2]=1;choice=1;board[1][2]='X';}
if ((go==7) && (pinakas[2][0]==0)) {pinakas[2][0]=1;choice=1;board[2][0]='X';}
if ((go==8) && (pinakas[2][1]==0)) {pinakas[2][1]=1;choice=1;board[2][1]='X';}
if ((go==9) && (pinakas[2][2]==0)) {pinakas[2][2]=1;choice=1;board[2][2]='X';}}

if (player==2){
if ((go==1) && (pinakas[0][0]==0)) {pinakas[0][0]=2;choice=1;board[0][0]='O';}
if ((go==2) && (pinakas[0][1]==0)) {pinakas[0][1]=2;choice=1;board[0][1]='O';}
if ((go==3) && (pinakas[0][2]==0)) {pinakas[0][2]=2;choice=1;board[0][2]='O';}
if ((go==4) && (pinakas[1][0]==0)) {pinakas[1][0]=2;choice=1;board[1][0]='O';}
if ((go==5) && (pinakas[1][1]==0)) {pinakas[1][1]=2;choice=1;board[1][1]='O';}
```

```
      if ((go==6) && (pinakas[1][2]==0)) {pinakas[1][2]=2;choice=1;board[1][2]='O';}
      if ((go==7) && (pinakas[2][0]==0)) {pinakas[2][0]=2;choice=1;board[2][0]='O';}
      if ((go==8) && (pinakas[2][1]==0)) {pinakas[2][1]=2;choice=1;board[2][1]='O';}
      if ((go==9) && (pinakas[2][2]==0)) {pinakas[2][2]=2;choice=1;board[2][2]='O';}}

     fp2=fopen("c:\\pinakas.txt","wb");
    for (i=0;i<3;i++){
    for (j=0;j<3;j++){
        fprintf(fp2,"%d\n",pinakas[i][j]);
          } }
      fclose(fp2);
   }while(go<1 || go>9 || choice!=1);

   /* Check for a winning line - diagonals first */
if (pinakas[0][0]!=0)
  {if (pinakas[0][0] == pinakas[1][1] && pinakas[1][1] == pinakas[2][2]) winner = player;}
if (pinakas[0][2]!=0)
  {if (pinakas[0][2] == pinakas[1][1] && pinakas[1][1] == pinakas[2][0]) winner = player;}
/* Check rows for a winning line */
if (pinakas[0][0]!=0)
  {if (pinakas[0][0] == pinakas[0][1] && pinakas[0][1] == pinakas[0][2]) winner = player;}
if (pinakas[1][0]!=0)
  {if (pinakas[1][0] == pinakas[1][1] && pinakas[1][1] == pinakas[1][2]) winner = player;}
if (pinakas[2][0]!=0)
  {if (pinakas[2][0] == pinakas[2][1] && pinakas[1][1] == pinakas[2][2]) winner = player;}
/* Check columns for a winning line */
if (pinakas[0][0]!=0)
  {if (pinakas[0][0] == pinakas[1][0] && pinakas[1][0] == pinakas[2][0]) winner = player;}
if (pinakas[0][1]!=0)
  {if (pinakas[0][1] == pinakas[1][1] && pinakas[1][1] == pinakas[2][1]) winner = player;}
if (pinakas[0][2]!=0)
```

```
  {if (pinakas[0][2] == pinakas[1][2] && pinakas[1][2] == pinakas[2][2]) winner = player;}

if(winner!=0)
  {
   //printf("\nCongratulations, player %d, YOU ARE THE WINNER!\n", winner);

  fp3=fopen("c:\\winner.txt","w");
     fprintf(fp3,"%d",winner);
  fclose(fp3);}
else
{
fp2=fopen("c:\\pinakas.txt","w");
  for (i=0;i<3;i++){
     //printf("\n");
   for (j=0;j<3;j++){ fprintf(fp2,"%d\n",pinakas[i][j]); //printf("%d   ",pinakas[i][j]); } }
fclose(fp2);}
if (winner==0)
{
 if (player==1)
    {
     player=2; fp1=fopen("c:\\player.txt","wb");fprintf(fp1,"%d",player);
    fclose(fp1);}
 else
    {
     player=1;fp1=fopen("c:\\player.txt","wb"); fprintf(fp1,"%d",player); fclose(fp1);}
  }
return(0); }
```