Ατομική Διπλωματική Εργασία

# ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΑΛΙΑΣ ΛΕΥΚΩΣΙΑΣ ΤΟΥ 19$^{ΟΥ}$ ΑΙΩΝΑ ΜΕ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟ ΤΡΟΠΟ

**Πάνος Τανελιάν**

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Μάιος 2011**

# ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

## ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ


# ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΠΑΛΙΑΣ ΛΕΥΚΩΣΙΑΣ ΤΟΥ 19$^{ΟΥ}$ ΑΙΩΝΑ ΜΕ ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟ ΤΡΟΠΟ


**Πάνος Τανελιάν**

Επιβλέπων Καθηγητής

Δρ. Γιώργος Χρυσάνθου



Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2011

# Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή μου κ. Γιώργο Χρυσάνθου για την πολύτιμη βοήθεια του και τις συνεχείς σημαντικές συμβουλές καθ όλη τη διάρκεια της υλοποίησης της διπλωματικής μου εργασίας.

Επίσης θα ήθελα να ευχαριστήσω την Εσπερία Ηλιάδη, τον διδακτορικό φοιτητή Σάββα Μιχαήλ και τους συμφοιτητές μου Χρίστο Σιακίδη, Μαρίνο Στυλιανού, Αντώνη Χατζηστάθη, Αντώνη Καρταπάνη, Δέσπω Καραγιώργη, Ανδρέα Παπαμιχαήλ και Γιώτα Γεωργίου για την συμπαράσταση και τις συμβουλές που μου παρείχαν.

# Περίληψη

Το θέμα της Ατομικής Διπλωματικής εργασίας με την οποία ασχολήθηκα είναι η αυτοματοποιημένη μοντελοποίηση πόλεων και συγκεκριμένα της παλιάς Λευκωσίας του 19$^{ου}$ αιώνα. Η δημιουργία πόλεων είναι πολύ σημαντική εργασία, και ειδικά παλιών πόλεων, αφού έχει πολλές εφαρμογές και προοπτική για ακόμη περισσότερες.

Με τον όρο αυτοματοποιημένη μοντελοποίηση, εννοούμε την μοντελοποίηση της πόλης με βάση κάποια αρχεία από την τότε εποχή στα οποία αναγράφονται κάποιες πληροφορίες για την κάθε ιδιοκτησία της περιοχής που θα μοντελοποιήσουμε.

Έχοντας λοιπών τους τίτλους ιδιοκτησίας της περιοχής σε ένα αρχείο, στόχος μας ήταν με τη βοήθεια αυτού του αρχείου και στη συνέχεια με τη βοήθεια κάποιου λογισμικού, να μοντελοποιήσουμε μια περιοχή της παλιάς Λευκωσίας. Άρα η διπλωματική μου εργασία μπορεί κάποιος να πει ότι χωρίζεται σε δυο κομμάτια. Το πρώτο κομμάτι είναι να η επεξεργασία του αρχείου και η απόκτηση κάποιων πληροφοριών από αυτό, έτσι ώστε να τις χρησιμοποιήσουμε στο δεύτερο κομμάτι, όπου είναι η δημιουργία της πόλης.

Στα πλαίσια της διπλωματικής μου εργασίας όμως, ασχολήθηκα μόνο με το πρώτο κομμάτι για το λόγο ότι συναντήσαμε πολλά προβλήματα. Τα προβλήματα αυτά αφορούσαν τα διάφορα δεδομένα εισόδου που είχαμε και θα τα δούμε αναλυτικά.

# Περιεχόμενα


Κεφάλαιο 1 ..... 1

Εισαγωγή ..... 1

1.1 Γενικά ..... 1

1.2 Πιθανές Χρήσεις Τρισδιάστατων Μοντέλων κτηρίων ..... 2

1.3 Περιγραφή Εργασίας ..... 3

1.4 Σκοπός και Κίνητρο Εργασίας ..... 4

1.5 Δομή Αναφοράς ..... 5

Κεφάλαιο 2 ..... 7

Προηγούμενη Δουλειά ..... 7

2.1 LiDAR ..... 7

2.2 Πόλεις από Δορυφόρους ..... 8

2.3 Προηγούμενες Εργασίες στο Πανεπιστήμιο Κύπρου ..... 9

2.4 Διαδικαστική Μοντελοποίηση Πόλεων ..... 12

2.4 L-Systems ..... 13

2.5 Σύστημα City Engine ..... 14

2.6 Εργασία Εσπερίας Ηλιάδη ..... 17

Κεφάλαιο 3 ..... 18

Επισκόπηση ..... 18

3.1 Γενικά ..... 18

3.2 Μεθοδολογία ..... 18

3.3 Δεδομένα Εισόδου ..... 21

3.4 Αλγόριθμοι που Χρησιμοποιήθηκαν ..... 22

3.5 Βιβλιοθήκη Open cv ..... 27

Κεφάλαιο 4 ..... 28

Υλοποίηση ..... 28

4.1 Γενικά ..... 28

4.2 Λογισμικά που Χρησιμοποιήθηκαν ..... 29

4.3 Υλοποίηση Μεθοδολογίας 1 ..... 31

4.4 Μεθοδολογία 2 ..... 43

Κεφάλαιο 5 ..... 60

Αποτελέσματα ..... 60

5.1 Γενικά ..... 60

5.2 Αποτελέσματα .....60
5.3 Συμπεράσματα .....63
Βιβλιογραφία .....67
Παράρτημα Α..... Α-1
Παράρτημα Β.....Β-1

# Κεφάλαιο 1

## Εισαγωγή


1.1 Γενικά....1

1.2 Πιθανές Χρήσεις Τρισδιάστατων Μοντέλων κτηρίων....2

1.2.1 Ταινίες....2
1.2.2 Ηλεκτρονικά Παιχνίδια....2
1.2.3 Αρχαιολογικούς Σκοπούς....2
1.2.4 Προσομοιωτές....3
1.2.5 Ακαδημαϊκούς Σκοπούς....3

1.3 Περιγραφή Εργασίας....3

1.4 Σκοπός και Κίνητρο Εργασίας....4

1.5 Δομή Αναφοράς....5


### 1.1 Γενικά

Τα γραφικά ηλεκτρονικών υπολογιστών εφαρμόζονται πλέον σε πολλές βιομηχανίες όπως τις ταινίες, τα ηλεκτρονικά παιχνίδια, τους διάφορους προσομοιωτές και σε πολλές άλλες. Μια από τις κύριες συνεισφορές των γραφικών είναι η τρισδιάστατη μοντελοποίηση κτηρίων και γενικά η τρισδιάστατη μοντελοποίηση ολόκληρων πόλεων. Αυτό γιατί είναι μια πολύπλοκη διαδικασία και χρειάζεται αρκετή λεπτομέρεια έτσι ώστε τα αποτελέσματα να είναι ικανοποιητικά. Έχουν γίνει αρκετές προσπάθειες για την καλύτερη δυνατή μοντελοποίηση πόλεων και έχουν αναπτυχθεί ειδικά λογισμικά που κάνουν αυτή τη δουλειά. Η τρισδιάστατη μοντελοποίηση πόλεων είναι πολύ σημαντική γιατί κάποιος μπορεί να δει και να ερευνήσει μια πόλη χωρίς να βρίσκεται σε αυτή. Ακόμη πιο σημαντική είναι η τρισδιάστατη μοντελοποίηση παλαιών πόλεων, που σήμερα δεν έχουμε πληροφορίες για αυτές, όπως για παράδειγμα μια σημερινή πόλη μπορούμε να έχουμε εικόνες από δορυφόρους ή και από πανοραμικά πλάνα. Αυτό είναι επίσης πολύ χρήσιμο γιατί θα μπορούμε να βλέπουμε μια πόλη σε τρισδιάστατη

μορφή, η οποία υπήρχε πιο παλιά αλλά πλέον δεν υπάρχει, και να καταλάβουμε αρχιτεκτονικές κτηρίων( και όχι μόνο) παλαιότερων πολιτισμών.

## 1.2 Πιθανές Χρήσεις Τρισδιάστατων Μοντέλων κτηρίων

### 1.2.1 Ταινίες

Μια πολύ σημαντική χρήση των μοντέλων αυτών είναι στις κινηματογραφικές ταινίες. Χρησιμοποιώντας εργαλεία που παράγουν τέτοια μοντέλα, ο χρήστης μπορεί να δημιουργήσει όσα κτίρια θέλει με ένα αρκετά αποδοτικό τρόπο. Αυτό είναι ένα μεγάλο πλεονέκτημα για την βιομηχανία των ταινιών γιατί μπορεί με αυτό τον τρόπο να δημιουργηθούν μεγάλες σκηνές με ένα μεσαίο-ικανοποιητικό επίπεδο λεπτομέρειας. Για τη δημιουργία πόλεων με μεγαλύτερη λεπτομέρεια, υπάρχουν περισσότερες απαιτήσεις από το σύστημα.

Σχήμα 1.1 Τρισδιάστατη πόλη σε ταινία

### 1.2.2 Ηλεκτρονικά Παιχνίδια

Η χρήση τέτοιου είδους λογισμικών δεν περιορίζεται μόνο στις κινηματογραφικές ταινίες. Ακόμη μια πολύ σημαντική χρήση των μοντέλων αυτών είναι στη βιομηχανία των ηλεκτρονικών υπολογιστών. Ένα από τα πιο σημαντικά βήματα που πρέπει να ακολουθηθούν για την ανάπτυξη ηλεκτρονικών παιχνιδιών, είναι η δημιουργία των μοντέλων που θα αποτελείται το παιχνίδι.

Σχήμα 1.2 Τρισδιάστατη πόλη σε ηλεκτρονικό παιχνίδι

Η μοντελοποίηση πόλεων σε παιχνίδια είναι πολύ σημαντική και συνάμα ακριβή διαδικασία. Υπάρχουν πολλά παιχνίδια όπου ο χρήστης κινείται και αλληλεπιδρά με την πόλη.

### 1.2.3 Αρχαιολογικούς Σκοπούς

Αυτά τα μοντέλα θα μπορούσαν επίσης να δώσουν μια εικονική πόλη η οποία θα μπορούσε να χρησιμοποιηθεί σε μουσεία και σε άλλα κοινωνικά-πολιτισμικά σημεία. Θα μπορούσε να χρησιμοποιηθεί για την βαθύτερη κατανόηση της αρχιτεκτονικής των

πόλεων από τους αρχαιολόγους. Επίσης Θα μπορούσε ο οποιοσδήποτε να δει τις αρχιτεκτονικές των αρχαίων πόλεων των ξένων χωρών που επισκέπτεται.

### 1.2.4 Προσομοιωτές

Το μοντέλο της πόλης θα μπορεί να χρησιμοποιηθεί για να τρέχουν διάφορες προσομοιώσεις όπως ροής αυτοκινήτων, ροής μάζας ανθρώπων, φυσικών καταστροφών, προσομοιώσεις για τον στρατό και την αστυνομία. Η προσομοίωση της ροής αυτοκινήτων θα είναι αρκετά χρήσιμη αφού μπορεί να χρησιμοποιηθεί για την αποφυγή συμφόρησης στους πολυσύχναστους δρόμους δίνοντας στον χρήστη εναλλακτικές διαδρομές για διάφορους προορισμούς. Επίσης θα μπορούσαν να χρησιμοποιηθούν για προσομοιώσεις από τον στρατό λαμβάνοντας διάφορες στρατηγικές καθώς επίσης και από την αστυνομία ελέγχοντας όλα τα πιθανά σενάρια για διάφορες κινήσεις που είναι πιθανόν να πράξει κάποιος εγκληματίας ανά πάσα στιγμή.

### 1.2.5 Ακαδημαϊκούς Σκοπούς

Τα μοντέλα αυτά των πόλεων θα μπορούσαν επίσης να χρησιμοποιηθούν και για εκπαιδευτικούς σκοπούς. Με αυτό τον τρόπο θα μπορούσαν οι εκπαιδευόμενοι, για να κατανοήσουν περισσότερο μια αρχαία πόλη και τον πολιτισμό της με το να εκπαιδεύεται με κάποιο εργαλείο το οποίο θα αναπαριστά την πόλη σε τρισδιάστατη μορφή.

## 1.3 Περιγραφή Εργασίας

Στόχος αυτής της διπλωματικής εργασίας είναι η αναπαράσταση μιας περιοχής της παλιάς Λευκωσίας του $19^{ου}$ αιώνα, ως ένα τρισδιάστατο μοντέλο με κάποιο αυτοματοποιημένο τρόπο. Αυτό όμως θα πρέπει να γίνει με βάση κάποιων πληροφοριών που έχουμε για την κάθε ιδιοκτησία της συγκεκριμένης περιοχής. Αυτές οι πληροφορίες είναι μια ηλεκτρονική μορφή των τίτλων ιδιοκτησίας της περιοχής, την εποχή εκείνη και αφορούν τον μοναδικό αριθμό της κάθε ιδιοκτησίας, τον ιδιοκτήτη ή τους ιδιοκτήτες, τους γείτονες της ιδιοκτησίας καθώς επίσης και κάποιες πληροφορίες

για το τι ήταν η ιδιοκτησία (κατάστημα, σπίτι κλπ) και πόσα τετραγωνικά ήταν. Με τη χρήση αυτών των πληροφοριών θα ακολουθηθούν κάποια βήματα τα οποία απαιτούνται για την τρισδιάστατη μοντελοποίηση της πόλης. Με τη βοήθεια αυτού του αρχείου θα πρέπει να κάνουμε τους απαραίτητους υπολογισμούς έτσι ώστε να δημιουργηθούν τα κτήρια. Δηλαδή, ο στόχος της εργασίας είναι η υλοποίηση κάποιου προγράμματος όπου παίρνει σαν παράμετρο αυτό το αρχείο και μετά την επεξεργασία του να μας δώσει κάποιες πληροφορίες όπως, πού θα κτιστεί το κάθε κτίριο, πόσο μεγάλο θα πρέπει να κτιστεί κλπ, και έπειτα με τη χρήση κάποιου ειδικού λογισμικού, να αναπαραστήσουμε τα κτίρια και συνεπώς τη παλιά Λευκωσία ως τρισδιάστατα μοντέλα. Στο αρχείο όμως υπήρχαν πολλά προβλήματα οποία θα δούμε στο κεφάλαιο 4, και αναγκαστήκαμε να ακολουθήσουμε μια άλλη μεθοδολογία, με άλλα δεδομένα εισόδου, έτσι ώστε να πάρουμε αυτές τις πληροφορίες.

### 1.4 Σκοπός και Κίνητρο Εργασίας

Διάφοροι παράγοντες αποδεικνύουν τον μεγάλο ρόλο που διαδραματίζουν στις μέρες μας η μοντελοποίηση πόλεων και γενικά η μοντελοποίηση κτηρίων και οι δυνατότητές τους στα πλαίσια της βιομηχανίας των ταινιών, των παιχνιδιών αλλά και στην εκπαίδευση.

Πολλές χώρες έχουν δώσει μεγάλη σημασία σε αυτό τον τομέα, δηλαδή, στην μοντελοποίηση των πόλεων για πολλούς λόγους. Ένας από τους κυριότερους είναι η στρατιωτική χρήση των μοντέλων αυτών. Αποτελούν ένα μέρος του αναλυτικού προγράμματος κάθε στρατιωτικής σχολής σε πολλές χώρες. Αυτό είναι πολύ χρήσιμο γιατί οι εκπαιδευόμενοι είναι σε θέση να γνωρίσουν μια πόλη και τα κατατόπια της, χωρίς να παραβρίσκονται σε αυτή.

Μια πιο μεγάλη πρόκληση είναι η μοντελοποίηση πιο παλιών πόλεων που δεν υπάρχουν πλέον και δεν μπορούμε να τις δημιουργήσουμε εύκολα. Ο σκοπός της διπλωματικής μου εργασίας είναι να μοντελοποιήσουμε μια περιοχή της παλιάς Λευκωσίας με ένα αυτοματοποιημένο τρόπο. Δηλαδή, όπως εξηγήσαμε πιο πάνω να γίνει η μοντελοποίηση της πόλης με βάση τους τίτλους ιδιοκτησίας που έχουμε για

εκείνη την περιοχή. Με την ανάπτυξη ενός τέτοιου προγράμματος θα δίνεται η δυνατότητα να μοντελοποιείται μια οποιαδήποτε παλιά πόλη φτάνει να έχουμε παρόμοιες πληροφορίες. Αυτό θα έχει ως αποτέλεσμα, αυτές οι πόλεις να χρησιμοποιούνται σε πολλές εφαρμογές. Η μεγαλύτερη χρησιμότητα ίσως αυτής της μοντελοποίησης είναι για πολιτισμικούς σκοπούς. Θα ήταν πολύ χρήσιμο στο να μαθαίνουμε για τους αρχαίους πολιτισμούς με αυτό τον ευχάριστο και εντυπωσιακό τρόπο.

Για την υλοποίηση της διπλωματικής μου εργασίας, λήφθηκαν υπόψη τα πιο πάνω, έτσι ώστε το πρόγραμμα να μπορεί να δημιουργεί μια οποιαδήποτε πόλη με τον ίδιο αυτοματοποιημένο τρόπο, με την προϋπόθεση να υπάρχουν οι επιθυμητές πληροφορίες.

### 1.5 Δομή Αναφοράς

Στην παρούσα Διπλωματική παρουσιάζεται η διαδικασία μοντελοποίησης μιας περιοχής της παλιάς Λευκωσίας και γίνεται αναφορά στα βήματα και τις μεθοδολογίες που ακολουθήθηκαν, καθώς και στα προβλήματα που συναντήθηκαν.

Στο δεύτερο Κεφάλαιο, Κεφάλαιο 2, γίνεται αναφορά στις προηγούμενες εργασίες που είχαν γίνει στο πανεπιστήμιο Κύπρου και αφορούσαν την μοντελοποίηση πόλεων. Γίνεται επίσης αναφορά στο λογισμικό city engine με το οποίο έχουν γίνει αρκετές μοντελοποιήσεις πόλεων με αποδοτικό τρόπο.

Στο τρίτο Κεφάλαιο, γίνεται μια επισκόπηση της εργασίας αναφέροντας τα βήματα που χρειάζονται για την ολοκλήρωση της εργασίας, τις δυο μεθοδολογίες που ακολουθήθηκαν , τα δεδομένα εισόδου που απαιτούνται καθώς επίσης και σε κάποιους αλγόριθμους οι οποίοι έπρεπε να ερευνηθούν για την ολοκλήρωση της εργασίας.

Στο Κεφάλαιο 4, παρουσιάζεται αναλυτικά, ο τρόπος με τον οποίο υλοποιήσαμε την εργασία. Αναλύουμε τα βήματα τα οποία ακολουθήθηκαν, τον τρόπο με τον οποίο τα υλοποιήσαμε, τα λογισμικά τα οποία χρησιμοποιήθηκαν ,τα προβλήματα που βρήκαμε καθ όλη τη διάρκεια της εργασίας καθώς επίσης θα αναφερθούμε στις λύσεις τις οποίες βρήκαμε στα πολλά προβλήματα που αντιμετωπίσαμε.

Τέλος, στο Κεφάλαιο 5, παρουσιάζονται και συζητούνται τα αποτελέσματα της εργασίας, τα συμπεράσματα στα οποία καταλήξαμε μετά την ολοκλήρωση της εργασίας καθώς επίσης και στις εμπειρίες που αποκτήθηκαν κατά την ολοκλήρωση της εργασίας και τέλος αναφέρονται κάποιες εισηγήσεις για περεταίρω διερεύνηση επί του θέματος.

# Κεφάλαιο 2

## Προηγούμενη Δουλειά

---

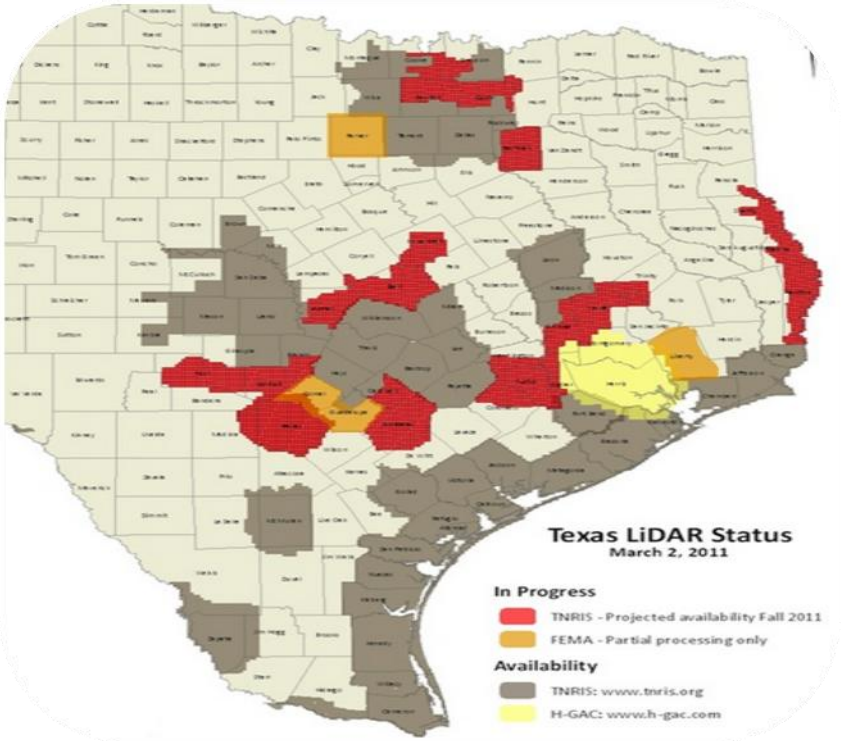
2.1 LiDAR .....7
2.2 Πόλεις από Δορυφόρους.....8
2.3 Προηγούμενες Εργασίες στο Πανεπιστήμιο Κύπρου .....9
2.3.1 Προηγούμενη Εργασία 1.....9
2.3.2 Προηγούμενη Εργασία 2.....10
2.3.3 Προηγούμενη Εργασία 3.....11
2.4 Διαδικαστική Μοντελοποίηση Πόλεων.....12
2.4 L-Systems .....13
2.5 Σύστημα City Engine.....14
2.6 Εργασία Εσπερίας Ηλιάδη.....17

---

### 2.1 LiDAR

Το LiDAR (Light Detection And Ranging) πρόκειται για μια οπτική τεχνολογία τηλεανίχνευσης (optical remote sensing technology) που μετρά ιδιότητες του διάχυτου φωτός για να βρει την απόσταση και κάποιες άλλες πληροφορίες κάποιου μακρινού στόχου. Η τεχνολογία LiDAR χρησιμοποιεί ένα αισθητήρα λέιζερ για να μεταδίδει και να λαμβάνει παλμούς από ανακλώμενες επιφάνειες. Τα δεδομένα LiDAR μπορούν να συλλέγονται κατά τη διάρκεια της μέρας ή της νύχτας, με την προϋπόθεση ότι το αεροπλάνο στο οποίο βρίσκεται η τεχνολογία να είναι κάτω από σύννεφα. Γνωρίζοντας την ακριβή θέση και τη στάση του αεροπλάνου στον αέρα, μπορούμε να υπολογίσουμε την απόσταση από το έδαφος και να υπολογίσουμε τις θέσεις των στοιχείων όπως (σπίτια, κτήρια, γήπεδα κλπ). Τα δεδομένα LiDAR μπορούν να επεξεργαστούν για να δημιουργηθεί μια γη η οποία δεν περιέχει ανθρωπογενές χαρακτηριστικά, που

αντιπροσωπεύει την επιφάνεια του εδάφους. Αυτή η μέθοδος όμως απαιτεί η πόλη να υπάρχει έτσι ώστε να πάρουμε τις πληροφορίες από αεροπλάνο.




Σχήμα 2.1: LiDAR πληροφορίες για το Texas

## 2.2 Πόλεις από Δορυφόρους

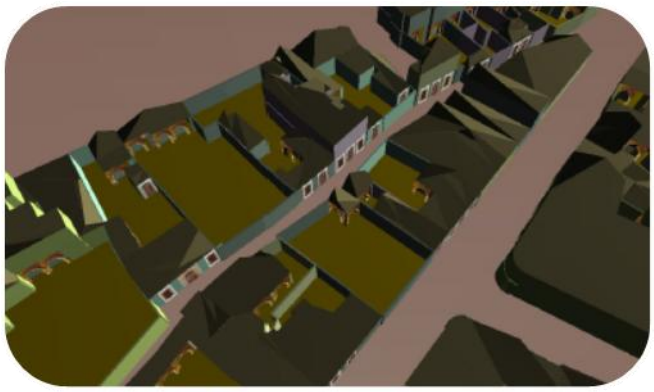
Στις μέρες μας η χρησιμοποίηση των δορυφόρων έχει γίνει μια πολύ σημαντική ανακάλυψη για τις ανεπτυγμένες χώρες για πολλούς σκοπούς όπως στρατιωτικούς, ερευνητικούς κλπ. Μια επίσης σημαντική χρήση των δορυφόρων είναι η αναπαράσταση πόλεων. Σήμερα, μπορεί κάποιος να ερευνήσει μια πόλη με τη χρήση κάποιου συστήματος και να δει πως ακριβώς αυτή φαίνεται. Αυτό αφορά το δεύτερο κομμάτι της δικής μου εργασίας, το κτίσιμο της πόλης, όμως, οι πληροφορίες παίρνονται από δορυφόρους και επίσης απαιτείται, όπως είναι λογικό, η πόλη να υπάρχει.


Σχήμα 2.2: Το πανεπιστήμιο Κύπρου απο δορυφόρο

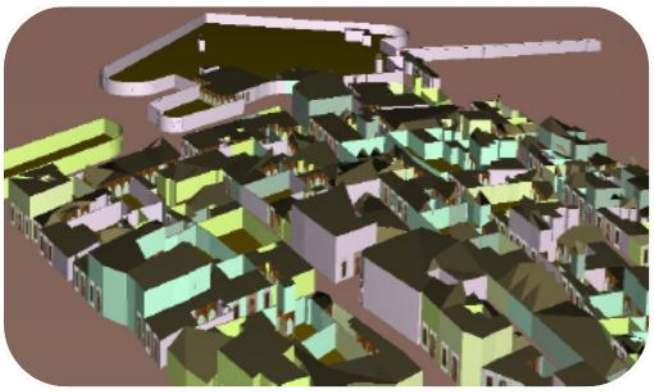
## 2.3 Προηγούμενες Εργασίες στο Πανεπιστήμιο Κύπρου

### 2.3.1 Προηγούμενη Εργασία 1

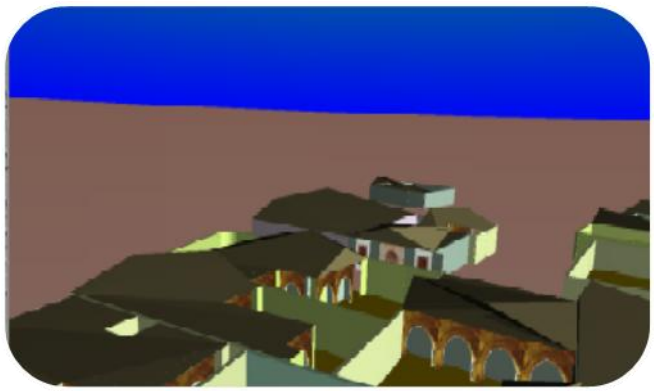
Αρχικά με αυτό το θέμα ασχολήθηκε η Μαριάννα Δικαιάκου το 2002. Μελέτησε την αρχιτεκτονική των κτιρίων της παλιάς Λευκωσίας και όρισε ένα αριθμό κανόνων για την παραγωγή μοντέλων. Ακολούθως δημιούργησε ένα πρόγραμμα χρησιμοποιώντας την γλώσσα προγραμματισμού Java, το οποίο έκτιζε τα κτίρια χωρίς να δημιουργείται η στέγη. Επίσης πρόσθεσε κάποιες έτοιμες εικόνες για να κάνουν το μοντέλο να φαίνεται ακόμη πιο ρεαλιστικό. Τα δεδομένα εισόδου ήταν ένας χάρτης της περιοχής ο οποίος αγοράστηκε από το Υπουργείο Εσωτερικών. Κάποια από τα αποτελέσματα της Μαριάννας φαίνονται πιο κάτω.



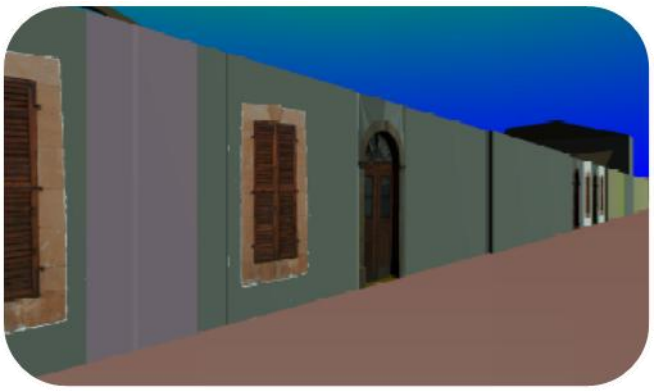
Σχήμα 2.3 Αποτελέσματα Μαριάννας Δικαιάκου



Σχήμα 2.4 Αποτελέσματα Μαριάννας Δικαιάκου



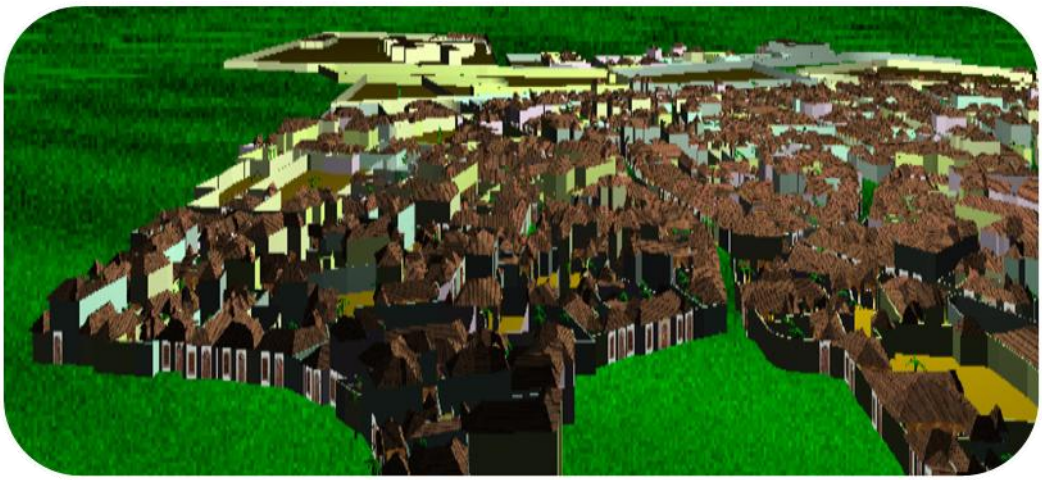
Σχήμα 2.5 Αποτελέσματα Μαριάννας Δικαιάκου



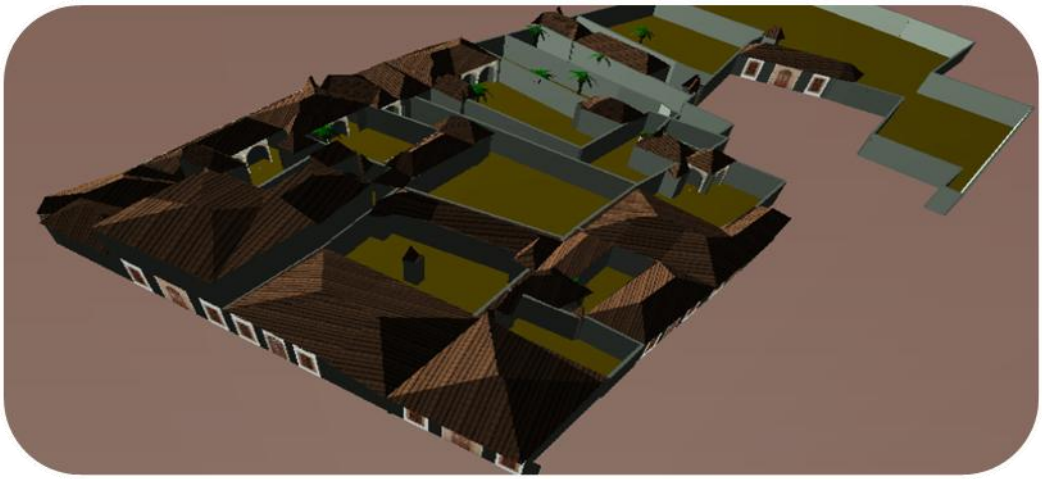
Σχήμα 2.6 Αποτελέσματα Μαριάννας Δικαιάκου

### 2.3.2 Προηγούμενη Εργασία 2

Την πιο πάνω εργασία συνέχισε ο Ανδρέας Ευθυμίου το 2003. Ο Ανδρέας είχε μελετήσει την δουλειά της Μαριάννας και πρόσθεσε ένα αλγόριθμο που σκοπό είχε να τοποθετηθούν οι στέγες στα σπίτια με κάποιο πιο ρεαλιστικό τρόπο. Ο αλγόριθμος αυτός (straight skeleton) δημιουργεί με βάση τα αρχικά αρχιτεκτονικά σχέδια, ένα νέο σκελετό πολυγώνων, το οποίο μπορεί να χρησιμοποιηθεί για την τρισδιάστατη κατασκευή της οροφής ενός κτιρίου. Κάποια από τα αποτελέσματα της εργασίας του Ανδρέα Ευθυμίου φαίνονται πιο κάτω.



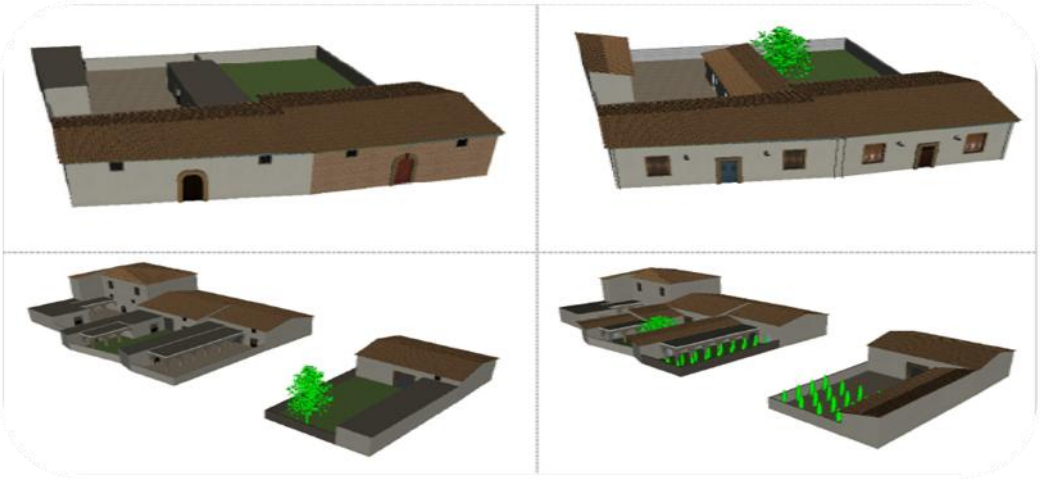
Σχήμα 2.7 Αποτελέσματα Ανδρέα Ευθυμίου



Σχήμα 2.8 Αποτελέσματα Ανδρέα Ευθυμίου

### 2.3.3 Προηγούμενη Εργασία 3

Τις δυο πιο πάνω εργασίες συνέχισε ο Σάββας Μιχαήλ το 2009. Ο Σάββας μελέτησε τις προηγούμενες εργασίες και χρησιμοποίησε το λογισμικό city engine για την αναπαράσταση της παλιάς Λευκωσίας του 19$^{ου}$ αιώνα. Με τη βοήθεια αυτού του λογισμικού, δημιούργησε κάποιους κανόνες όπου με βάση αυτούς τους κανόνες το λογισμικό έκτισε την πόλη. Είχαν δημιουργηθεί αρκετά σπίτια και επομένως αρκετά πολύγωνα και τα αποτελέσματα φαίνονταν σχετικά γρήγορα(10 σπίτια ~1 λεπτό) με την προϋπόθεση να χρησιμοποιείται ένας δυνατός υπολογιστής. Κάποια από τα αποτελέσματα της εργασίας του Σάββα Μιχαήλ φαίνονται πιο κάτω.



Σχήμα 2.9 Αποτελέσματα Σάββα Μιχαήλ

Σχήμα 2.10 Αποτελέσματα Σάββα Μιχαήλ

## 2.4 Διαδικαστική Μοντελοποίηση Πόλεων

Η μοντελοποίηση μιας πόλης έχει ως αποτέλεσμα έναν αριθμό προβλημάτων στα γραφικά υπολογιστών. Κάθε αστική περιοχή διαθέτει ένα δίκτυο μεταφοράς που ακολουθεί την εξέλιξη του πληθυσμού και τις περιβαλλοντικές επιδράσεις. Τα παραγόμενα κτίρια ακολουθούν ιστορικούς κανόνες, κανόνες αισθητικής και υποχρεωτικούς κανόνες. Για να δημιουργηθεί μια εικονική πόλη, πρέπει να σχεδιαστεί ένας οδικός χάρτης και ένας μεγάλος αριθμός κτιρίων πρέπει να παραχθεί.

Η CGA(Computer Graphics Architecture) shape, είναι μια γραμματική σχημάτων για διαδικαστική μοντελοποίηση της αρχιτεκτονικής γραφικών υπολογιστών, παράγοντας σπίτια με υψηλή ποιότητα και γεωμετρική λεπτομέρεια. Μπορεί να παράξει εκτεταμένα αρχιτεκτονικά μοντέλα για ηλεκτρονικά παιχνίδια και ταινίες σε χαμηλά κόστη.

Σχήμα 2.11: Κτήρια παραγώμενα απο την CGA

Η δημιουργία συναρπαστικών μοντέλων είναι μια σημαντική εργασία στην δημιουργία επιτυχημένων ταινιών και ηλεκτρονικών μοντέλων. Όμως, η μοντελοποίηση μεγάλων τρισδιάστατων περιβαλλόντων, όπως για παράδειγμα πόλεις, είναι μια πολύ ακριβή διαδικασία και μπορεί να χρειαστεί αρκετά ανθρωποχρόνια. Ο τρόπος ο οποίος χρησιμοποιεί το λογισμικό City Engine είναι η διαδικαστική μοντελοποίηση

χρησιμοποιώντας γραμματική σχημάτων, ικανή για την δημιουργία μεγάλων πόλεων με υψηλή γεωμετρική λεπτομέρεια, αποδοτικά και γρήγορα. Η χρησιμοποίηση της CGA shape γραμματικής με κανόνες παραγωγής, εξελίσσουν επαναληπτικά κάποιο σχέδιο, δημιουργώντας σε κάθε επανάληψη, περισσότερη λεπτομέρεια. Οι κανόνες παραγωγής, αρχικά δημιουργούν ένα ακατέργαστο ογκώδες μοντέλο και στη συνέχεια κτίζεται η πρόσοψη. Στο τέλος προσθέτεται περισσότερη λεπτομέρεια όπως για παράδειγμα για τα παράθυρα, τις πόρτες κ.τ.λ. Το μεγάλο πλεονέκτημα της μεθόδου αυτής, είναι ότι η επαναχρησιμοποίηση των κανόνων με μικρές διαφοροποιήσεις μπορεί να προκαλέσει τη δημιουργία μεγάλων διαφορετικών αρχιτεκτονικών, φτιάχνοντας έτσι μια ολόκληρη πόλη. Οι κανόνες αυτοί συνήθως είναι παραλλαγές των κανόνων L-systems.

## 2.4 L-Systems

Τα L-system (Lindenmayer system) είναι ένα παράλληλο σύστημα κανόνων, δηλαδή μια παραλλαγή μιας κανονικής γραμματικής (ένα σύνολο κανόνων και συμβόλων ), τα οποία είναι γνωστά για την χρησιμοποίηση τους για την μοντελοποίηση της διαδικασίας ανάπτυξης φυτών *(σχήμα 2.12)*. Είναι όμως επίσης ικανό να μοντελοποιήσει τη μορφολογία ποικίλων οργανισμών. Τα L-systems μπορούν επίσης να χρησιμοποιηθούν για την παραγωγή όμοιων κλασμάτων όπως τα επαναληπτικά συστήματα . Τα L-systems αναπτύχθηκαν από τον Ούγγρο βιολόγο από το πανεπιστήμιο του Utrecht, Aristid Lindenmayer (1925-1989).

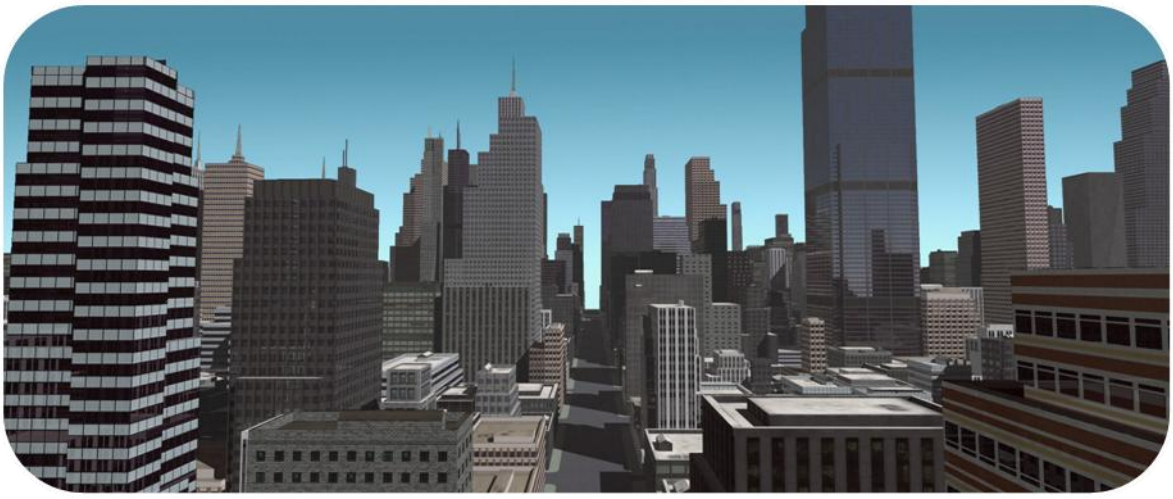
Η αναδρομική φύση των κανόνων των L-systems οδηγεί στην ομοιότητα και στην κλασματική μορφή(fractal like form) η οποία είναι εύκολο να περιγραφή με ένα L σύστημα. Τα μοντέλα φυτών και οι φαινομενικά φυσικές οργανικές μορφές μπορούν επίσης να οριστούν εύκολα, αυξάνοντας το αναδρομικό επίπεδο όπου η μορφή αναπτύσσεται αργά και γίνεται ακόμα πιο πολύπλοκη. Τα συστήματα αυτά είναι επίσης γνωστά για την παραγωγή τεχνητής ζωής.

Οι κανόνες των L-systems εφαρμόζονται επαναληπτικά αρχίζοντας από την αρχική κατάσταση. Η διαφορά μεταξύ των L-systems και κανονικών γλωσσών είναι ότι στα L systems μπορούν να εφαρμοστούν πολλοί κανόνες ταυτόχρονα σε κάθε
επανάληψη. Αν οι κανόνες εφαρμόζονταν μόνο ένας κάθε φορά, τότε θα παραγόταν μια γλώσσα και όχι ένα L σύστημα.

Σχήμα 2.12: Δέντρα δημιουργημένα χρησιμοποιώντας τα L-systems σε τρισδιάστατη μορφή

## 2.5 Σύστημα City Engine

Το city engine είναι ένα λογισμικό που έχει τη δυνατότητα να δημιουργήσει πολύπλοκα μοντέλα πόλεων, γρήγορα και αποδοτικά και είναι φιλικό προς τον χρήστη. Για αυτό το λόγο έχουν αναπαρασταθεί με αυτό το λογισμικό πολλές πόλεις και όπως προαναφέραμε έχουν πολλές εφαρμογές. Μια πόλη αποτελείται από οδικούς χάρτες, από τα κτίρια και από textures πρόσοψης. Αυτό που κάνει τη χρήση του city engine τόσο σημαντική, είναι το ότι οι πόλεις είναι πολύ δύσκολο να αναπαρασταθούν χειροκίνητα. Για αυτό το λόγο, για την μοντελοποίηση πολύπλοκων περιβάλλων χρησιμοποιούνται διαδικαστικές μέθοδοι. Στα επόμενα σχήματα φαίνονται κάποια παραδείγματα από αποτελέσματα του λογισμικού



Σχήμα 2.13 Manhattan-3D

Σχήμα 2.14 Zurich-3D

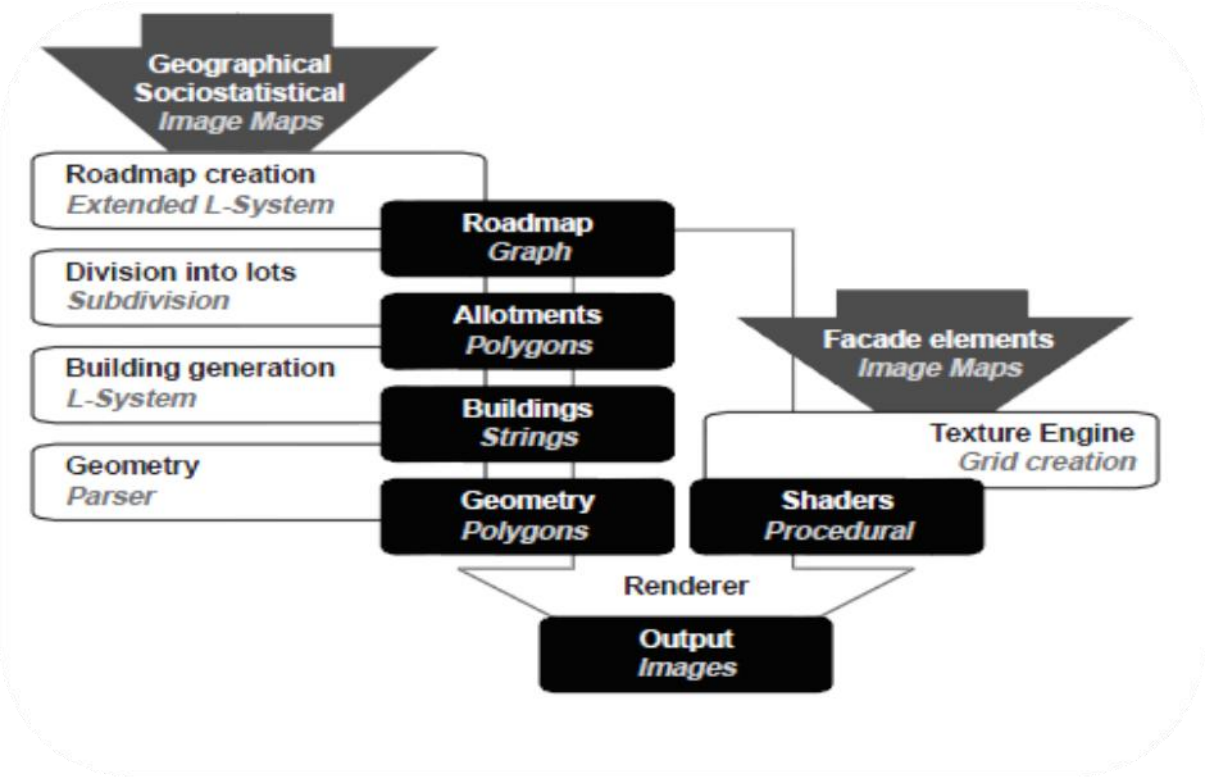
Το σύστημα City Engine αποτελείται από πολλά διαφορετικά εργαλεία που αποτελούν το pipeline που φαίνεται στο σχήμα 2.5.

Σε ένα πρώτο στάδιο, τα δεδομένα εισόδου τροφοδοτούνται στο σύστημα παραγωγής γενιάς, με τη χρήση ενός επιτεταμένου L-συστήματος. Τα δεδομένα εισόδου στο πρώτο στάδιο είναι μια εικόνα ενός χάρτη και οι παράμετροι για τους κανόνες.

Στη συνέχεια, οι περιοχές μεταξύ των δρόμων υποδιαιρούνται για να καθοριστούν οι τοποθεσίες στις οποίες θα τοποθετηθούν τα κτήρια. Σε αυτή τη φάση, σαν είσοδο έχουμε τον γράφο των δρόμων και ένα χάρτη της περιοχής χρήσης, ενώ σαν έξοδο παίρνουμε ένα σύνολο πολυγώνων από τις τοποθεσίες στις οποίες θα τοποθετηθούν τα κτίρια.

Στην τρίτη φάση, εφαρμόζοντας ένα άλλο L-σύστημα, τα κτήρια παράγονται ως μια string αναπαράσταση των Boolean πράξεων σε απλά στερεά σχήματα. Σε αυτή τη φάση, έχουμε σαν είσοδο το σύνολο πολυγώνων που πήραμε από την Τρίτη φάση και έχουμε σαν έξοδο μια σειρά κτηρίων με επιπρόσθετες πληροφορίες.

Τέλος, ένα πρόγραμμα ανάλυσης ερμηνεύει όλα τα αποτελέσματα για το λογισμικό απεικόνισης. Το λογισμικό απεικόνισης θα πρέπει να είναι σε θέση να επεξεργάζεται πολυγωνική γεωμετρία και texture χάρτες. Αυτό ισχύει για σχεδόν οποιοδήποτε 3D renderer. Επιπλέον, οι πιο πολλοί scanline renderers υποστήριξη διαδικαστικά textures, έτσι ώστε ο προτεινόμενος μηχανισμός για την παραγωγή προσόψεων των κτιρίων να μπορεί να ενσωματωθεί στο pipeline.




Σχήμα 2.15 Το pipeline του συστήματος city engine για την μοντελοποίηση πόλεων

### 2.6 Εργασία Εσπερίας Ηλιάδη

Οι πιο πάνω εργασίες που είδαμε αφορούσαν μόνο την μοντελοποίηση πόλεων με διαφορετικούς τρόπους. Η Εσπερία, στα πλαίσια του διδακτορικού της έχει ασχοληθεί με το πρώτο κομμάτι της δικής μας εργασίας. Δηλαδή, στο να αποκτήσει πληροφορίες για την τότε εποχή με δεδομένα που πήρε από το κτηματολόγιο και τα μετέτρεψε σε αρχείο κειμένου. Είχε υπολογίσει με τα δεδομένα εκείνα τις ομάδες γειτόνων και τα όρια ιδιοκτησιών, όπου είναι και το πρώτο κομμάτι της δικής μας διπλωματικής. Αυτό όμως το έχει πετύχει χειροκίνητα, δηλαδή, δεν είχε υλοποιήσει κάποιο πρόγραμμα που να βρίσκει αυτές τις πληροφορίες όπως στην δική μας περίπτωση. Η εργασία της όμως επικεντρωνόταν στην αρχιτεκτονική των κτηρίων της τότε εποχής και όχι στον ακριβές υπολογισμό των ορίων ιδιοκτησιών και των ομάδων γειτόνων.

Η δική μας διπλωματική εργασία, αφορά την αναπαράσταση μιας παλιάς πόλης (της παλιάς Λευκωσίας), όπου δεν έχουμε πολλές πληροφορίες σε ηλεκτρονική μορφή για την τότε εποχή. Θα προσπαθήσουμε όμως με δεδομένα που έχουμε, τους τίτλους ιδιοκτησιών της περιοχής, να αναπαραστήσουμε την πόλη με ένα αυτοματοποιημένο τρόπο και όχι χειροκίνητα. Οι προηγούμενες δουλειές που είχαν γίνει, αφορούσαν την διαδικαστική αναπαράσταση των πόλεων και έδιναν έμφαση στην λεπτομέρεια των κτηρίων και στην αποδοτικότητα του συστήματος. Στη δική μας εργασία προσπαθήσαμε να το κάνουμε αυτό με ένα διαφορετικό τρόπο. θέλουμε να ορίσουμε με τη βοήθεια ενός προγράμματος που πρέπει να υλοποιήσουμε, το πώς και πού ήταν κτισμένο το κάθε σπίτι, έτσι ώστε στη συνέχεια με τη βοήθεια του λογισμικού που είδαμε πιο πάνω να αναπαραστήσουμε μια περιοχή της παλιάς Λευκωσίας σε τρισδιάστατη μορφή.

# Κεφάλαιο 3

## Επισκόπηση

---

3.1 Γενικά .....18
3.2 Μεθοδολογία .....18
3.3 Δεδομένα Εισόδου .....21
3.4 Αλγόριθμοι που Χρησιμοποιήθηκαν .....22
3.4.1 Αλγόριθμος Flood fill .....22
3.4.2 Αλγόριθμος Corner Detection .....24
3.4.3 Αλγόριθμος Image Thinning .....25
3.5 Βιβλιοθήκη Open cv .....27

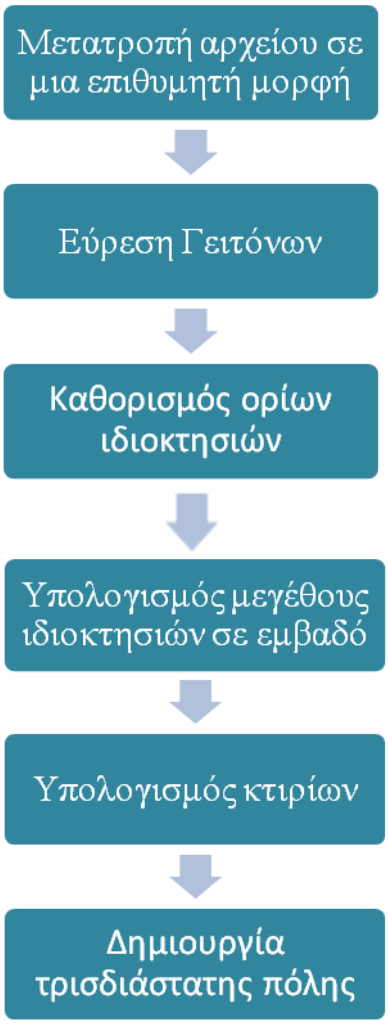
---

### 3.1 Γενικά

Στο τρίτο κεφάλαιο γίνεται μια επισκόπηση της ατομικής διπλωματικής εργασίας. Αναλύουμε τα βήματα που πρέπει να ακολουθηθούν για την ολοκλήρωση της εργασίας και στη συνέχεια γίνεται μια αναφορά σε αλγόριθμους και δεδομένα εισόδου που χρησιμοποιήσαμε. Τέλος, αναφερόμαστε στην βιβλιοθήκη open cv.

### 3.2 Μεθοδολογία

Όπως αναφέραμε και στα προηγούμενα κεφάλαια, ο στόχος της εργασίας είναι να μοντελοποιήσουμε μια περιοχή της παλιάς Λευκωσίας, σε τρισδιάστατη μορφή με κάποιο αυτοματοποιημένο τρόπο και όχι όπως είδαμε σε προηγούμενες εργασίες. Για να πετύχουμε αυτό το στόχο απαιτούνται κάποια βήματα που πρέπει να γίνουν. Αφού λοιπόν δεν έχουμε εικόνες της Λευκωσίας της εποχής εκείνης, θα πρέπει να κτίσουμε την πόλη με ένα διαφορετικό τρόπο. Πιο κάτω φαίνονται σχηματικά τα βήματα που πρέπει να ακολουθηθούν.




Τώρα ας δούμε τα βήματα ένα-ένα πιο αναλυτικά. Κατά αρχή, για να μπορέσουμε να φτάσουμε στην μοντελοποίηση της πόλης θα πρέπει να έχουμε κάποιες πληροφορίες για αυτή. Για να μπορέσουμε να επεξεργαστούμε τις πληροφορίες θα πρέπει να είναι σε μια μορφή ευκολοδιάβαστη από ένα πρόγραμμα. Άρα στο πρώτο βήμα, θα πρέπει να μετατρέψουμε, τις πληροφορίες που έχουμε σαν είσοδο, σε μια μορφή η οποία να μας βοηθά, έτσι ώστε η επεξεργασία τους να είναι όσο πιο αποδοτική γίνεται. Σε αυτό το βήμα λοιπόν σαν είσοδο έχουμε ένα αρχείο κειμένου όπου σε αυτό αναγράφονται οι τίτλοι ιδιοκτησίας της περιοχής. Όμως, ένα πρόγραμμα και γενικά ένα σύστημα δεν είναι εύκολο να επεξεργαστεί δεδομένα από ένα αρχείο κειμένου. Άρα το πρώτο βήμα χρειάζεται για να έχουμε τις πληροφορίες για τις ιδιοκτησίες σε μια πιο οργανωμένη

μορφή. Όταν εφαρμοστεί το πρώτο βήμα, θα έχουμε σαν έξοδο ένα αρχείο το οποίο θα περιέχει τους τίτλους ιδιοκτησίας, σε μια ευκολοδιάβαστη μορφή από ένα πρόγραμμα. Αφού εφαρμοστεί το πρώτο βήμα, θα πρέπει στη συνέχεια να ορίσουμε την τοποθεσία του κάθε σπιτιού. Δηλαδή, θα πρέπει με βάση το αρχείο που πήραμε σαν έξοδο από το πρώτο βήμα, να ανακαλύψουμε το που βρίσκεται το κάθε σπίτι. Για να γίνει όμως αυτό, θα πρέπει πρώτα να ορίσουμε ποιες ιδιοκτησίες ανήκουν στην κάθε περιοχή. Δηλαδή να βρούμε τις ομάδες γειτόνων. Άρα σε αυτό το δεύτερο βήμα θα πάρουμε σαν είσοδο το αρχείο που πήραμε σαν έξοδο από το πρώτο βήμα και μετά την επεξεργασία του αρχείου να έχουμε σαν έξοδο τις ομάδες γειτόνων. Το δεύτερο βήμα χρειάζεται γιατί, για να μοντελοποιήσουμε σωστά την πόλη, θα πρέπει να τοποθετήσουμε την κάθε ιδιοκτησία εκεί που πρέπει. Εάν υπολογίσουμε τις ομάδες γειτόνων, στη συνέχεια θα μπορέσουμε να βρούμε και όλες τις πληροφορίες που χρειάζονται για να ορίσουμε σωστά την τοποθεσία και το μέγεθος της κάθε ιδιοκτησίας.

Στο τρίτο βήμα, αφού τώρα θα γνωρίζουμε τις ομάδες γειτόνων, θα πρέπει να υπολογίσουμε τα όρια των ιδιοκτησιών. Δηλαδή σε κάθε περιοχή, να χωρίσουμε τις ιδιοκτησίες όπως ήταν αυτές χωρισμένες την εποχή εκείνη. Αυτό πρέπει να γίνει για να υπολογίσουμε ξεχωριστά το μέγεθος της κάθε ιδιοκτησίας αλλά επίσης και να τοποθετήσουμε σωστά την κάθε ιδιοκτησία στην περιοχή της. Δηλαδή, σε αυτό το βήμα θα έχουμε σαν είσοδο το αρχείο που πήραμε από το πρώτο βήμα αλλά επίσης και τα αποτελέσματα του δεύτερου βήματος που έχουμε τις ομάδες γειτόνων, και με το τέλος του τρίτου βήματος θα έχουμε την πληροφορία για το πώς είναι οριοθετημένες οι ιδιοκτησίες σε κάθε περιοχή.

Αφού αποκτήσουμε αυτές τις πληροφορίες στα προηγούμενα βήματα, στο τέταρτο βήμα θα πρέπει να υπολογίσουμε το μέγεθος της κάθε ιδιοκτησίας. Αυτό μπορεί να γίνει με το να υπολογίσουμε το εμβαδό της κάθε περιοχής ιδιοκτησιών και στη συνέχεια της κάθε ιδιοκτησίας. Αυτό χρειάζεται γιατί, στη συνέχεια όταν θα πρέπει να μοντελοποιήσουμε τα κτήρια, θα πρέπει να γνωρίζουμε το μέγεθος που είχαν. Δηλαδή, σαν είσοδο στο τέταρτο βήμα θα έχουμε τις πληροφορίες από τα προηγούμενα βήματα(γείτονες, όρια ιδιοκτησιών και το αρχείο με τους τίτλους ιδιοκτησίας) και σαν έξοδο θα έχουμε με κάποιο τρόπο το εμβαδό της κάθε ιδιοκτησίας.

Στο πέμπτο βήμα, αφού βρήκαμε το που και πόσο μεγάλα θα κτιστούν τα κτήρια, θα πρέπει να βρούμε το πώς ήταν το κάθε κτήριο. Δηλαδή, θα πρέπει να δούμε εάν ήταν σπίτι, καφενείο, κατάστημα κλπ, για να ξέρουμε με ποιο τρόπο θα κτιστεί. Αυτή η πληροφορία βρίσκεται στο αρχείο που πήραμε σαν έξοδο από το πρώτο βήμα. Άρα στην είσοδο αυτού του βήματος θα έχουμε το πιο πάνω αρχείο και σαν έξοδο θα έχουμε οργανωμένες πληροφορίες για το πώς ήταν κτισμένο το κάθε κτήριο.

Στο τελευταίο βήμα θα πρέπει να γίνει η μοντελοποίηση σε τρισδιάστατη μορφή της πόλης. Αφού θα έχουμε όλα τα πιο πάνω στοιχεία, τότε θα μπορούμε να δημιουργήσουμε την πόλη, όπου είναι και το ζητούμενο της εργασίας. Για να το πετύχουμε αυτό θα πρέπει να γράψομε κάποιους κανόνες σε κάποιο λογισμικό για να δημιουργηθεί η πόλη. Για να το κάνουμε αυτό θα πρέπει να λάβουμε υπόψη μας τις διάφορες αρχιτεκτονικές των κτηρίων της τότε εποχής.

Στην υλοποίηση, δεν ακολουθήθηκαν όλα τα πιο πάνω βήματα όπως περιγράφονται πιο πάνω. Ο λόγος είναι ότι στο αρχείο με τους τίτλους ιδιοκτησίας υπήρχαν πολλά προβλήματα τα οποία δεν μπορούσαμε να τα παραλείψουμε. Έτσι για αυτό, ακολουθήσαμε άλλες μεθόδους για να βρούμε τις πληροφορίες που χρειαζόμασταν. Τα προβλήματα στο αρχείο θα αναλυθούν μαζί με όλα τα άλλα προβλήματα που συναντήσαμε καθ όλη τη διάρκεια της εργασίας. Για αυτό το λόγο προσπαθήσαμε να υλοποιήσουμε την εργασία με διαφορετικά δεδομένα εισόδου και ως αποτέλεσμα να ακολουθήσουμε διαφορετική μεθοδολογία. Στο επόμενο Κεφάλαιο , Κεφάλαιο 4 θα δούμε αναλυτικά και τις δυο μεθοδολογίες που ακολουθήσαμε.

### 3.3 Δεδομένα Εισόδου

Τα αρχικά δεδομένα εισόδου, όπως αναφέραμε και στο κεφάλαιο 1, ήταν ένα .αρχείο κειμένου(.doc) όπου σε αυτό υπήρχαν καταχωρημένες διάφορες πληροφορίες για την κάθε ιδιοκτησία οι οποίες ήταν απαραίτητες για την κατασκευή των κτιρίων της παλιάς Λευκωσίας. Αυτό το αρχείο, το οποίο είναι μια ηλεκτρονική μορφή των τίτλων ιδιοκτησίας της συγκεκριμένης περιοχής, περιέχει τις εξής πληροφορίες για τις 227 ιδιοκτησίες της περιοχής της Χρυσαλινιώτισσας : Έναν αύξον αριθμό για την κάθε ιδιοκτησία, το σχέδιο στο οποίο βρίσκεται(sheet/plan), το οικόπεδο (Το σχέδιο και το

οικόπεδο μπορούν να χαρακτηρίσουν μοναδικά μια ιδιοκτησία ), τον ιδιοκτήτη της ιδιοκτησίας, τους γείτονες της ιδιοκτησίας, καθώς επίσης και κάποιες άλλες πληροφορίες (όπου σε κάθε περίπτωση μπορούν να διαφέρουν) όπως το τι κτίριο είναι, από πόσους ορόφους αποτελείται, εάν έχει αυλή και συνήθως αναγράφεται από πόσα τετραγωνικά μέτρα αποτελείται.

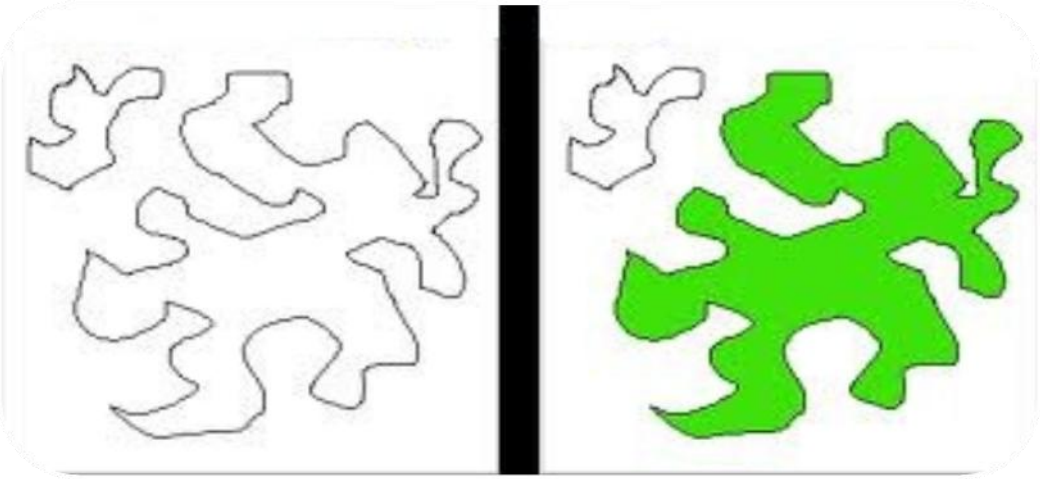
Αφού παρατηρήσαμε τα πολλά προβλήματα τα οποία προαναφέραμε, χρειάστηκε να ψάξουμε άλλα δεδομένα εισόδου. Αφού μιλήσαμε με την Εσπερία της ζητήσαμε εάν είχε τον χάρτη της περιοχής σε κάποια ηλεκτρονική μορφή. Τελικά αυτό που καταφέραμε να βρούμε είναι τον χάρτη της περιοχής σε .dwg(drawing) μορφή στον οποίο υπήρχαν επίσης και τα όρια των ιδιοκτησιών. Αυτό που θέλαμε να κάνουμε όμως από το σχέδιο αυτό ήταν να εξάγουμε με κάποιο τρόπο τις κορυφές(σε συντεταγμένες χ - ψ) των τεμαχίων τα οποία χωρίζονται από τα όρια των ιδιοκτησιών. Αφού εξάγαμε τις κορυφές με τη βοήθεια κάποιου λογισμικού σε μορφή .obj, είδαμε ότι τα τεμάχια δεν ήταν σχεδόν καθόλου οργανωμένα και έτσι δεν μπορέσαμε να χρησιμοποιήσουμε το συγκεκριμένο obj αρχείο.

Μετά από αυτό αποφασίσαμε να εξάγουμε τα όρια των ιδιοκτησιών σε μορφή εικόνας έτσι ώστε να δίνεται σαν είσοδος σε ένα πρόγραμμα το οποίο θα υπολογίζει τις κορυφές με διαφορετικό τρόπο.

### 3.4 Αλγόριθμοι που Χρησιμοποιήθηκαν

#### 3.4.1 Αλγόριθμος Flood fill

Ο αλγόριθμος Flood fill**,** επίσης γνωστός και ως seed fill, εφαρμόζεται στον τομέα της επεξεργασίας εικόνας. Παίρνει σαν είσοδο μια εικόνα, ένα εικονοστοιχείο(pixel) σε συντεταγμένες χ-ψ, ένα χρώμα στόχου(target color) και ένα χρώμα γεμίσματος(fill color) και ξεκινά από το pixel το οποίο παίρνει σαν είσοδο και προχωρά προς όλες τις κατευθύνσεις(τέσσερεις η οκτώ ανάλογα με την κάθε περίπτωση) και γεμίζει όλα τα pixels με το χρώμα γεμίσματος μέχρι να φτάσει στο χρώμα στόχου. Ένα παράδειγμα εφαρμογής του αλγορίθμου φαίνεται πιο κάτω.



Σχήμα 3.1: Παράδειγμα εφαρμογής Flood Fill

Μπορούμε να δούμε στην εικόνα στα αριστερά δυο σχήματα τα οποία είναι άδεια στο εσωτερικό τους. Στην εικόνα στα δεξιά βλέπουμε το μεγάλο σχήμα να γεμίζεται από πράσινα pixels. Αυτό σημαίνει ότι εφαρμόστηκε στην εικόνα το flood fill και είχε σαν είσοδο ένα pixel το οποίο βρισκόταν κάπου μέσα στο μεγάλο σχήμα. Το χρώμα γεμίσματος ήταν το πράσινο και το χρώμα στόχου ήταν το μαύρο (το χρώμα των σχημάτων).

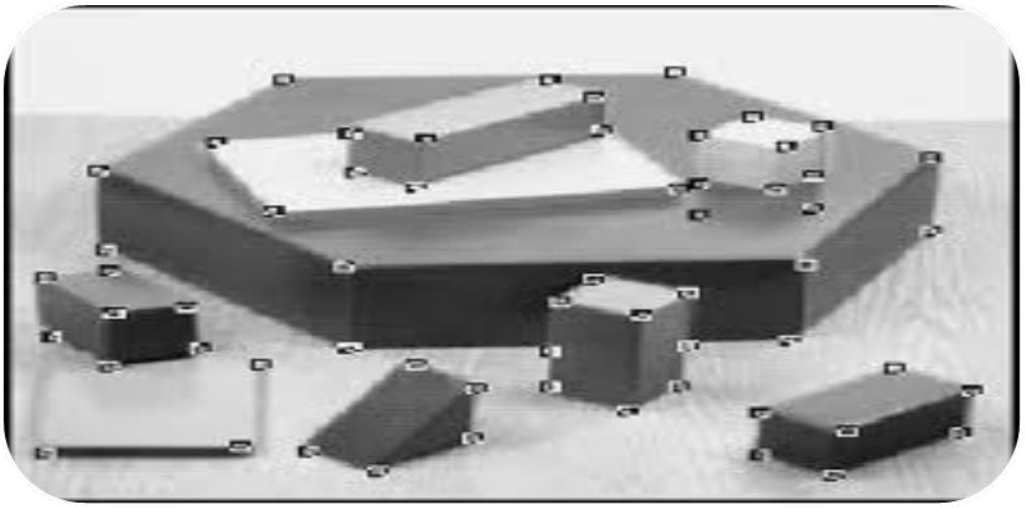
Για την υλοποίηση του αλγόριθμου μπορεί να χρησιμοποιηθεί αναδρομή ή ακόμη και στοίβα γιατί σε εικόνες με πολλά pixels δεν μπορεί να χρησιμοποιηθεί αναδρομή για λόγους μνήμης και αποδοτικότητας. Ένα παράδειγμα ψευδοκώδικα που υλοποιεί flood fill με χρήση αναδρομής φαίνεται πιο κάτω.

```
Flood-fill (node, target-color, replacement-color):
 1. If the color of node is not equal to target-color, return.
 2. Set the color of node to replacement-color.
 3. Perform Flood-fill (one step to the west of node, target-color,
replacement-color).
    Perform Flood-fill (one step to the east of node, target-color,
replacement-color).
    Perform Flood-fill (one step to the north of node, target-color,
replacement-color).
    Perform Flood-fill (one step to the south of node, target-color,
replacement-color).
 4. Return
```

### 3.4.2 Αλγόριθμος Corner Detection

Το corner detection(ανίχνευση γωνιών) είναι μια λύση ενός κοινού προβλήματος που χρησιμοποιείται μέσα στα συστήματα υπολογιστικής όρασης για να εξαγάγει ορισμένα είδη χαρακτηριστικών γνωρισμάτων και να συμπεράνει το περιεχόμενο μιας εικόνας. Η ανίχνευση γωνιών χρησιμοποιείται συχνά σε πολλούς τομείς όπως το motion detection, το image matching, το image mosaicin, το 3D modeling και το object recognition.

Με δεδομένης μια εικόνας το corner detection ανιχνεύει και επιστρέφει τις γωνιές που υπάρχουν στην εικόνα. Ανάλογα με το τι θέλει ο χρήστης μπορεί να ρυθμίσει διάφορες παραμέτρους(αυτό βεβαία εξαρτάται από την κάθε υλοποίηση του corner detection) έτσι ώστε να ορίσει αυτός το τι είναι μια γωνιά. Με αυτό τον τρόπο ο χρήστης μπορεί να πάρει διαφορετικά αποτελέσματα ανάλογα με τις παραμέτρους που θα θέσει. Στην πιο κάτω εικόνα βλέπουμε ένα παράδειγμα εφαρμογής του αλγόριθμου corner detection

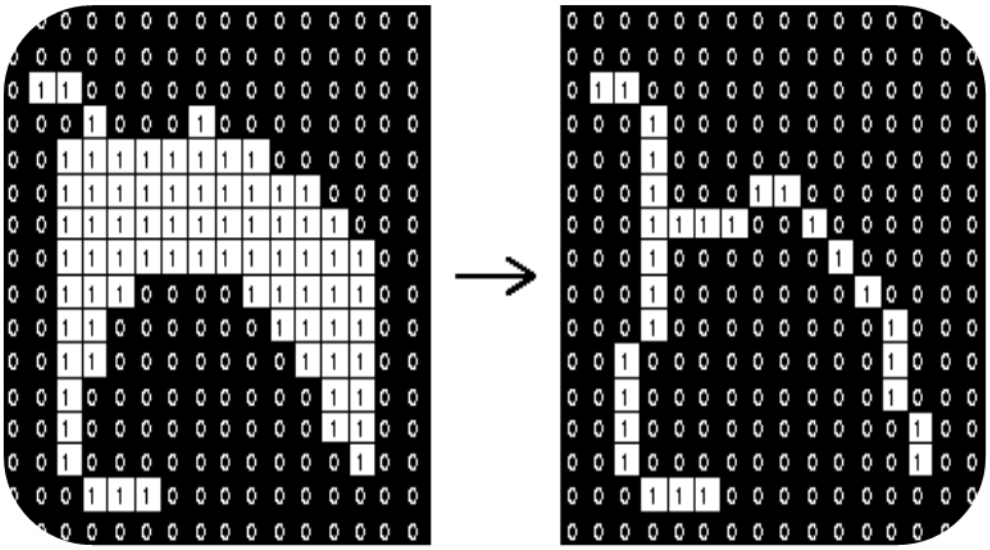


Σχήμα 3.2: Παράδειγμα εφαρμογής corner detection

Υπάρχουν αρκετοί αλγόριθμοι που υλοποιούν το corner detection και κάποιοι από αυτούς είναι ενσωματωμένοι σε βιβλιοθήκες που χρησιμοποιούνται για την επεξεργασία εικόνας. Παραδείγματα τέτοιων αλγορίθμων είναι το Harris Corner Detection και το Shi-Tomasi Corner Detection.

### 3.4.3 Αλγόριθμος Image Thinning

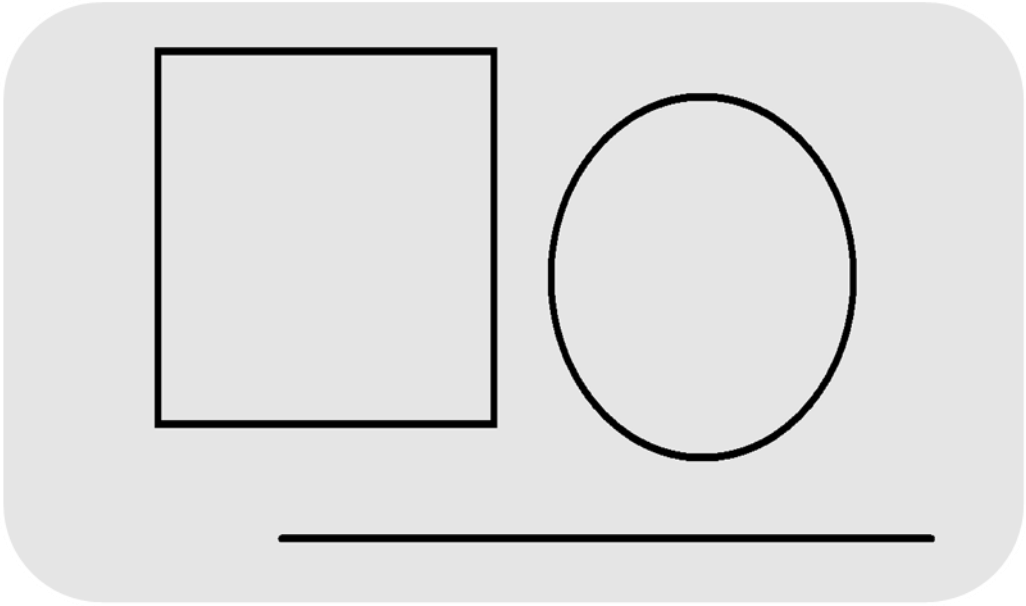
Ο αλγόριθμος image thinning χρησιμοποιείται στην επεξεργασία εικόνας. Είναι μια μορφολογική λειτουργία που χρησιμοποιείται για να αφαιρέσει τα επιλεγμένα pixels έτσι ώστε οι γραμμές της εικόνας να γίνουν πιο λεπτές. Μπορεί να χρησιμοποιηθεί για διάφορες εφαρμογές, αλλά είναι ιδιαίτερα χρήσιμο για το skeletonization.



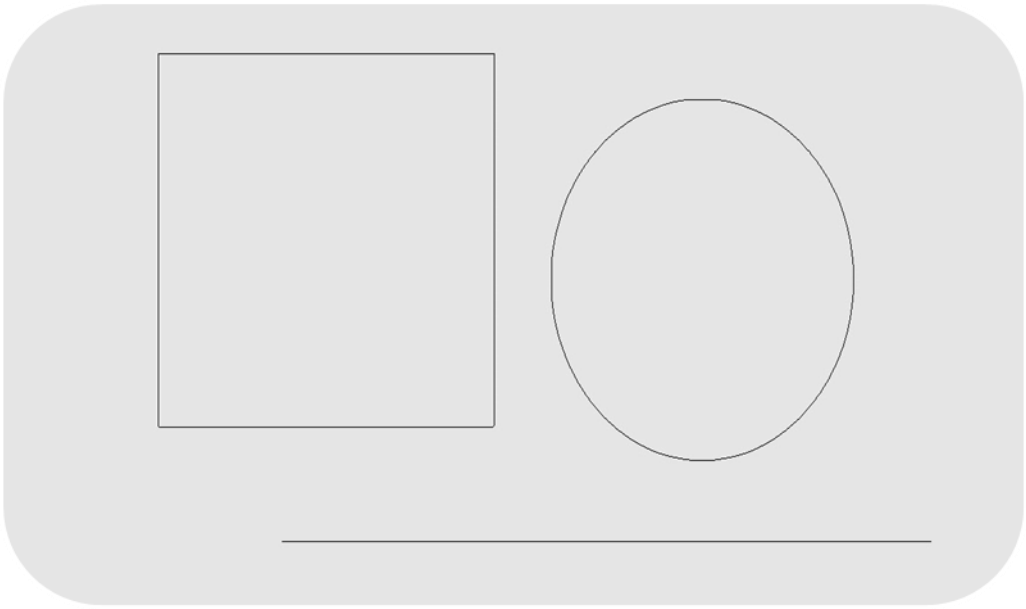
Σχήμα 3.3: Παράδειγμα εφαρμογής image thinning

Σε αυτόν τον τρόπο χρησιμοποιείται συνήθως για να μειώσει το πάχος όλων των γραμμών σε ενιαίο πάχος εικονοστοιχείου(pixel). Ο αλγόριθμος αυτός κανονικά εφαρμόζεται μόνο στις δυαδικές εικόνες. Παίρνει σαν είσοδο μια εικόνα και μετά την επεξεργασία της μας επιστρέφει σαν έξοδο μια νέα εικόνα με όλες τις γραμμές της να έχουν πάχος μόνο ενός pixel.

Στην εικόνα 3.3 βλέπουμε την αρχική εικόνα εισόδου και στην εικόνα 3.4 την εικόνα που μας δίνει σαν έξοδο το image thinning με την συγκεκριμένη είσοδο.



Σχήμα 3.4: Εικόνα εισόδου image thinnig



Σχήμα 3.5: Εικόνα εξόδου image thinning

Μπορούμε εύκολα να παρατηρήσουμε ότι στην πρώτη εικόνα οι γραμμές αποτελούνται από πολλά pixels, ενώ στην δεύτερη, όταν εφαρμόσουμε image thinning, βλέπουμε ότι οι γραμμές αποτελούνται από μόνο ένα pixel.

## 3.5 Βιβλιοθήκη Open cv

Η open cv (Open Computer Vision Library) είναι μια βιβλιοθήκη από συναρτήσεις όπου υλοποιούν περισσότερους από 500 αλγόριθμους που αφορούν την υπολογιστική όραση και αναπτύχθηκε από την Intel. Είναι δωρεάν στη χρήση όσο για ακαδημαϊκούς τόσο και για βιομηχανικούς σκοπούς. Επικεντρώνεται κυρίως στην επεξεργασία εικόνας σε πραγματικό χρόνο. Αρχικά είχε υλοποιηθεί στην γλώσσα προγραμματισμού C, αλλά έχει μια πλήρης διπροσωπία και έγιναν όλες οι βελτιστοποιήσεις στην C++.

Ασχολείται με αρκετά προβλήματα όπου αφορούν την υπολογιστική όραση όπως object recognition, object tracking, face recognition, corner detection, feature detection, edge detection, motion tracking και σε άλλα πολλά. Αναφερόμαστε στην βιβλιοθήκη αυτή γιατί χρησιμοποιήθηκε για την υλοποίηση της εργασίας.

# Κεφάλαιο 4

## Υλοποίηση

---

4.1 Γενικά .....28

4.2 Λογισμικά που Χρησιμοποιήθηκαν .....29

4.2.1 Microsoft Excel .....29
4.2.2 AutoCAD 2011 .....30
4.2.3 3D Studio Max 9.0 .....30
4.2.4 Eclipse .....30
4.2.5 Visual studio 2008 .....30

4.3 Υλοποίηση Μεθοδολογίας 1 .....31

4.3.1 Ανάλυση Μεθοδολογίας 1 .....31
4.3.2 Προβλήματα Μεθοδολογίας 1 .....34

4.4 Μεθοδολογία 2 .....43

---

### 4.1 Γενικά

Το μεγαλύτερο μέρος της εργασίας είχε να κάνει σχέση με την υλοποίηση. Ο αρχικός μας στόχος ήταν να υλοποιήσουμε ένα πρόγραμμα το οποίο θα έπαιρνε σαν είσοδο το αρχείο που μετατρέψαμε από .doc μορφή σε .csv μορφή, το οποίο όπως αναφέραμε και σε προηγούμενα κεφάλαια περιείχε κάποιες πληροφορίες για την κάθε ιδιοκτησία, και να μας δίνει σαν έξοδο κάποιες άλλες, πιο οργανωμένες πληροφορίες όπως ποιες ήταν οι ομάδες γειτόνων, ποια ήταν τα όρια των ιδιοκτησιών, το πώς ήταν κτισμένο το κτίριο, εάν είχε αυλή κλπ. Αυτές τις πληροφορίες θα τις χρειαζόμασταν για να τις δώσουμε σε μεταγενέστερο στάδιο στο λογισμικό City Engine, έτσι ώστε να δημιουργήσουμε τα κτίρια με βάση αυτές τις πληροφορίες που θα παίρναμε σαν έξοδο από το πρόγραμμα που υλοποιήσαμε. Δηλαδή θέλαμε να κτίσουμε την περιοχή της Χρυσαλινιότισσας της παλιάς Λευκωσίας με ένα αυτοματοποιημένο τρόπο και όχι όπως

γινόταν αυτό τα προηγούμενα χρόνια σε παλαιότερες διπλωματικές. Στο συγκεκριμένο αρχείο όμως παρουσιάζονταν αρκετά προβλήματα. Κάποια από αυτά καταφέραμε να τα λύσουμε (εύκολα ή δύσκολα) και κάποια όμως από αυτά δεν μπορέσαμε να τα λύσουμε γιατί υπήρχαν πολλές ασάφειες στο αρχείο. Αυτά τα προβλήματα θα τα δούμε εκτενέστερα σε αυτό το κεφάλαιο ένα προς ένα, καθώς επίσης και τις λύσεις που έχουμε βρει. Αφού παρατηρήσαμε αυτά τα προβλήματα απευθυνθήκαμε στην Εσπερία για να μας δώσει(εάν είχε) τον χάρτη της περιοχής σε κάποια ηλεκτρονική μορφή. Τελικά μας είχε δώσει ένα σχέδιο(αρχείο της μορφής dwg.) της συγκεκριμένης περιοχής. Το ανοίξαμε στο λογισμικό AutoCAD για να δούμε πως θα μπορέσουμε να το χρησιμοποιήσουμε στην δική μας περίπτωση. Παρατηρήσαμε ότι από αυτό το αρχείο μπορούσαμε να εξάγουμε(export) τις κορυφές των τεμαχίων σε συντεταγμένες χ-ψ. Αυτό γιατί σε αυτό το σχέδιο υπήρχαν τα όρια των ιδιοκτησιών, δηλαδή το πώς ήταν χωρισμένες οι ιδιοκτησίες και έτσι κάνοντας export τις κορυφές των ορίων ιδιοκτησίας να υλοποιήσουμε ένα άλλο πρόγραμμα το οποίο με βάση αυτές τις κορυφές να υπολογίζει τους γείτονες. Δυστυχώς όμως και σε αυτή την περίπτωση υπήρχαν αρκετά προβλήματα τα οποία θα δούμε σε αυτό το κεφάλαιο, και έτσι δεν μπορέσαμε ούτε με αυτό τον τρόπο να βρούμε αυτό που αρχικά θέσαμε σαν στόχο, τον υπολογισμό των γειτόνων. Μετά από αυτό, αποφασίσαμε να εξάγουμε τα όρια των ιδιοκτησιών από το σχέδιο σαν μια εικόνα (και όχι σαν κορυφές) και επεξεργάζοντας αυτή την εικόνα να υπολογίσουμε με αυτό τον τρόπο τις κορυφές. Σε αυτό το κεφάλαιο θα δούμε εκτενέστερα τις μεθοδολογίες που χρησιμοποιήσαμε, όλα τα προβλήματα που είχαμε μαζί και με τις λύσεις που βρήκαμε σε αρκετά από αυτά, τους αλγόριθμους που υλοποιήσαμε στη συνέχεια σε σχέση με την επεξεργασία της εικόνας και γενικά όλη τη διαδικασία που ακολουθήσαμε στην υλοποίηση.

## 4.2 Λογισμικά που Χρησιμοποιήθηκαν

### 4.2.1 Microsoft Excel

Το λογισμικό αυτό χρειάστηκε για να μετατρέψουμε τα αρχικά μας δεδομένα από .doc μορφή σε .csv μορφή. Αυτή η μετατροπή χρειάστηκε να γίνει γιατί εάν είχαμε τα αρχικά δεδομένα εισόδου τότε θα ήταν δύσκολο να τα επεξεργαστούμε σωστά και

αποδοτικά. Κάνοντας αυτή τη μετατροπή μπορούσαμε να δώσουμε το .csv αρχείο σαν είσοδο στο πρόγραμμα και να επεξεργαστούμε τα δεδομένα σωστά και αποδοτικά.

### 4.2.2 AutoCAD 2011

Αυτό το λογισμικό χρειάστηκε για να ανοίξουμε και να επεξεργαστούμε το .dwg αρχείο. Το συγκεκριμένο αρχείο είχε γίνει στο ίδιο λογισμικό (autoCAD) αλλά σε μια παλαιότερη έκδοση και έτσι για να δούμε ακριβώς τι πληροφορίες περιείχε έπρεπε να το ανοίξουμε με το συγκεκριμένο λογισμικό. Στη συνέχεια το χρησιμοποιήσαμε για να εξάγουμε σε μορφή .pdf τα όρια των ιδιοκτησιών έτσι ώστε να μπορέσουμε στη συνέχεια να τα επεξεργαστούμε σαν μια εικόνα.

### 4.2.3 3D Studio Max 9.0

Χρησιμοποιήθηκε για να εξάγουμε τις κορυφές των ορίων των ιδιοκτησιών σε συντεταγμένες, έτσι ώστε να τις δώσουμε σαν είσοδο σε ένα πρόγραμμα το οποίο θα βρίσκει βάση αυτών των κορυφών τους γείτονες και στη συνέχεια τα εμβαδά της κάθε περιοχής.

### 4.2.4 Eclipse

Αυτό το λογισμικό χρειάστηκε για να αναπτύξουμε το πρόγραμμα στην γλώσσα προγραμματισμού java όπου παίρνει σαν είσοδο το αρχείο για τις ιδιοκτησίες και υπολογίζει τις ομάδες γειτόνων της περιοχής. Επιλέγηκε αυτό το λογισμικό για την ανάπτυξη του προγράμματος γιατί ήμουν ήδη εξοικειωμένος με αυτό.

### 4.2.5 Visual studio 2008

Χρησιμοποιήθηκε για την ανάπτυξη ενός άλλου προγράμματος το οποίο παίρνει σαν είσοδο μια binary εικόνα και υπολογίζει τις κορυφές της κάθε ιδιοκτησίας. Για την επεξεργασία της εικόνας χρησιμοποιήθηκε η βιβλιοθήκη opencv αφού ήμουν ήδη εξοικειωμένος με την συγκεκριμένη βιβλιοθήκη. Η εικόνα που παίρνει σαν είσοδο είναι τα όρια των ιδιοκτησιών της περιοχής της παλιάς Λευκωσίας όπου πήραμε σαν έξοδο από το λογισμικό autoCAD και περιείχε μόνο τα όρια των ιδιοκτησιών (στο

αρχικό .dwg αρχείο που μας έχει δώσει η Εσπερία υπήρχαν πολλές άλλες-αχρείαστες για εμάς πληροφορίες και έτσι εξάγαμε μόνο τα όρια των ιδιοκτησιών).

### 4.3 Υλοποίηση Μεθοδολογίας 1

#### 4.3.1 Ανάλυση Μεθοδολογίας 1

Για την υλοποίηση της διπλωματικής εργασίας χρησιμοποιήθηκαν δυο διαφορετικές μεθοδολογίες ανεξάρτητες μεταξύ τους. Αυτό γιατί όπως προαναφέραμε υπήρξαν πολλά προβλήματα με τα δεδομένα εισόδου. Η πρώτη μεθοδολογία αφορά την αρχική προσέγγιση της εργασίας και έχει να κάνει σχέση με τα αρχικά δεδομένα εισόδου.

Στόχος ήταν να αναπτυχθεί ένα πρόγραμμα στην γλώσσα προγραμματισμού java το οποίο με είσοδο ένα αρχείο txt να υπολογίζει τις ομάδες γειτόνων της περιοχής, τα όρια των ιδιοκτησιών, τα εμβαδά της κάθε περιοχής, και το πώς ήταν χτισμένο το κάθε κτίριο. Το δεδομένο εισόδου το οποίο μας έχει δοθεί σε μορφή .doc, έπρεπε να μετατραπεί σε μορφή .csv και έπειτα σε μορφή txt. Μετά αφού κάναμε αυτές τις μετατροπές ήμασταν έτοιμοι να υλοποιήσουμε το πρόγραμμα.

Το συγκεκριμένο αρχείο περιείχε πληροφορίες για 255 ιδιοκτησίες της περιοχής. Πιο κάτω φαίνονται οι πρώτες πέντε καταχωρήσεις στο αρχείο για την καλύτερη κατανόηση της μεθοδολογίας του προγράμματος:

1 16 120 Church of Chrysaliniotissa "street, Maria Athanasi, Church of Chrysaliniotissa, river & street" "coffee- house of three rooms, one verandah & garden = two evleks + 620 sq ft"

2 16 121 Maria Athanasi wife of Costi Soteriou Parperi "street, Kallou Stavrino & Church of Chrysaliniotissa" "house of two rooms, one verandah, kitchen, yard & tree = [1100 sq ft]"

3 16 122 Kallou Stavrinis wife of Athanasi Kassabi "street, Styliani Avraami, Church of Chrysaliniotissa & Maria Athanasi" house of two rooms & right of passage through plot 121 = [430 sq ft]

4 16 123 Styliani Avraami "street, Church of Chrysaliniotissa & …….. & Kallou Stavrini " "house of three rooms, one verandah, one kitchen & yard & tree = [2500 sq ft]"

5 16 124 "Joannis Kyriakides, Sofoclis HjChristofi, Christofis Jacovaki,Yeorgios Pastellides & Costis Karayeorghi as trustees of Chrysaliniotissa Church Nicosia" Styliani Avraami "garden of two rooms, one stable & verandah & wheel well & tank = [eight donums, one evlek & 1725 sqft]"

Η πρώτη πληροφορία που είναι καταχωρημένη στο αρχείο είναι ένας αύξον αριθμός της κάθε ιδιοκτησίας ο οποίος ξεκινά από το 1 και τελειώνει μέχρι και το 255 όπου είναι και ο αριθμός των ιδιοκτησιών που είναι καταχωρημένες στο αρχείο. Στη συνέχεια ακολουθούν το σχέδιο το οποίο βρίσκεται η ιδιοκτησία (sheet) και το οικόπεδο. Η επόμενη πληροφορία που έχουμε είναι ο ιδιοκτήτης της ιδιοκτησίας. Σε πολλές περιπτώσεις μια ιδιοκτησία είχε πολλούς ιδιοκτήτες. Στη συνέχεια έχουμε τους γείτονες της κάθε ιδιοκτησίας. Παρατηρούμε ότι αναγράφονται τα ονόματα του κάθε γείτονα και όχι ο αριθμός της ιδιοκτησίας. Αυτό πολλές φορές δημιουργεί πρόβλημα για το λόγο ότι κάποιος μπορεί να έχει πολλές ιδιοκτησίες. Τέλος ακολουθούν πληροφορίες όσον αφορά την τοπολογία του σπιτιού. Δηλαδή το τι είδους κτίριο είναι (σπίτι, κατάστημα, καφενείο κλπ) και τον αριθμό των τετραγωνικών που αποτελείται.

Για την επαλήθευση των δικών μας αποτελεσμάτων, πήραμε από την Εσπερία μια εικόνα του χάρτη της περιοχής στην οποία φαίνονταν οι αριθμοί των ιδιοκτησιών και τα όρια των ιδιοκτησιών ήταν χωρισμένα. Με αυτό τον τρόπο μπορούσαμε να δούμε πως η Εσπερία βρήκε τις ομάδες γειτόνων και να τις συγκρίνουμε με τα δικά μας αποτελέσματα. Στο επόμενο σχήμα φαίνεται ο χάρτης της περιοχής με τα αποτελέσματα της Εσπερίας.

Σχήμα 4.1 Χάρτης περιοχής που φαίνονται τα όρια ιδιοκτησιών

Παρατηρούμε επίσης ότι οι πέντε πρώτες εγγραφές ανήκουν στην ίδια ομάδα γειτόνων. Δηλαδή, και οι πέντε ιδιοκτησίες έχουν τουλάχιστον ένα κοινό γείτονα με τις υπόλοιπες τέσσερεις.

Αφού στόχος μας λοιπόν ήταν με βάση αυτές τις πληροφορίες να βρούμε τις ομάδες γειτόνων, Το πρόγραμμα που αναπτύξαμε διαβάζει το αρχείο μια φορά και το φορτώνει σε ένα πίνακα δυο διαστάσεων έτσι ώστε να μην χρειαστεί να το διαβάσουμε περισσότερες από μια φορές. Το επόμενο βήμα ήταν να αποφασίσουμε πως θα αποθηκεύαμε τις ομάδες των γειτόνων που θα υπολογίζαμε και καταλήξαμε να τις αποθηκεύσουμε σε vector. Πιο κάτω φαίνεται η δήλωση του vector που κρατά τις ομάδες γειτόνων.

```
static private HashMap<String, HashSet<String>> omades = new HashMap<String,
HashSet<String>>();
```

Κάθε εγγραφή, δηλαδή κάθε ομάδα, έχει ένα αριθμό και όλους τους ιδιοκτήτες από όπου αποτελείται.

Η αρχική ιδέα ήταν να διαβάσουμε τον δυσδιάστατο πίνακα και για κάθε ιδιοκτησία να ελέγχουμε τους γείτονές της. Εάν κάποιος από τους γείτονες της ανήκει σε μια ιδιοκτησία, τότε και αυτή η ιδιοκτησία θα τοποθετηθεί σε αυτή την ομάδα. Εάν όχι, τότε δημιουργείται μια νέα ομάδα. Πιο κάτω φαίνεται ο αρχικός αλγόριθμος ο οποίος έχει υλοποιηθεί.

```
Foreach entry in the table
        If(omades.isempty)
                createNewTeam;
        if(some of its neighbors is a part of some group)
                addThisEntry to the same group
        else
                createNewTeam;
```

Εφαρμόζοντας όμως αυτό τον αλγόριθμο δεν είχαμε τα αποτελέσματα που θέλαμε. Όταν παρατηρήσαμε τα αποτελέσματα προσέξαμε ότι δεν ήταν καθόλου σωστά σε σχέση με τα αποτελέσματα που βρήκε η Εσπερία. Αυτό οφείλεται στα διάφορα προβλήματα και στις διάφορες ασάφειες που υπήρχαν στο αρχείο εισόδου. Στο επόμενο υποκεφάλαιο θα παρουσιάσουμε και θα εξηγήσουμε όλα τα προβλήματα που βρήκαμε στο αρχείο μαζί και με τις λύσεις, σε όποια είχαν.

### 4.3.2 Προβλήματα Μεθοδολογίας 1

#### 4.3.2.1 Πρόβλημα 1

Είναι εμφανές ότι για να αποφασίσουμε εάν μια ιδιοκτησία ανήκει σε κάποια ομάδα γειτόνων τότε θα πρέπει να κάνουμε σύγκριση συμβολοσειρών. Το πρώτο πρόβλημα αφορά τα ονόματα των ιδιοκτήτων. Πολλές φορές ένα όνομα αναγραφόταν διαφορετικά σε δυο διαφορετικές περιπτώσεις. Δηλαδή σε μια εγγραφή κάποιας ιδιοκτησίας ο ιδιοκτήτης γραφόταν με διαφορετικό τρόπο από αυτόν που γραφόταν ο ίδιος ιδιοκτήτης σε κάποιο γείτονά του και έτσι το πρόγραμμα θεωρούσε τα δυο αυτά strings

διαφορετικά αφού ήταν όντως διαφορετικά γραμμένα. Το μέγεθος του προβλήματος ήταν πολύ μεγάλο αφού ο ίδιος άνθρωπος γραφόταν διαφορετικά σε διαφορετικές περιπτώσεις.

Η αρχική προσέγγιση στο πρόβλημα ήταν όταν ελέγχουμε δυο strings να δούμε εάν είναι τα ίδια, σε περίπτωση που δεν είναι, τότε να ελέγχουμε χαρακτήρα-χαρακτήρα, πόσους χαρακτήρες διαφέρουν. Αυτό όμως δεν είναι σωστό. Ένα απλό παράδειγμα δείχνει πως αυτή η προσέγγιση είναι λάθος.

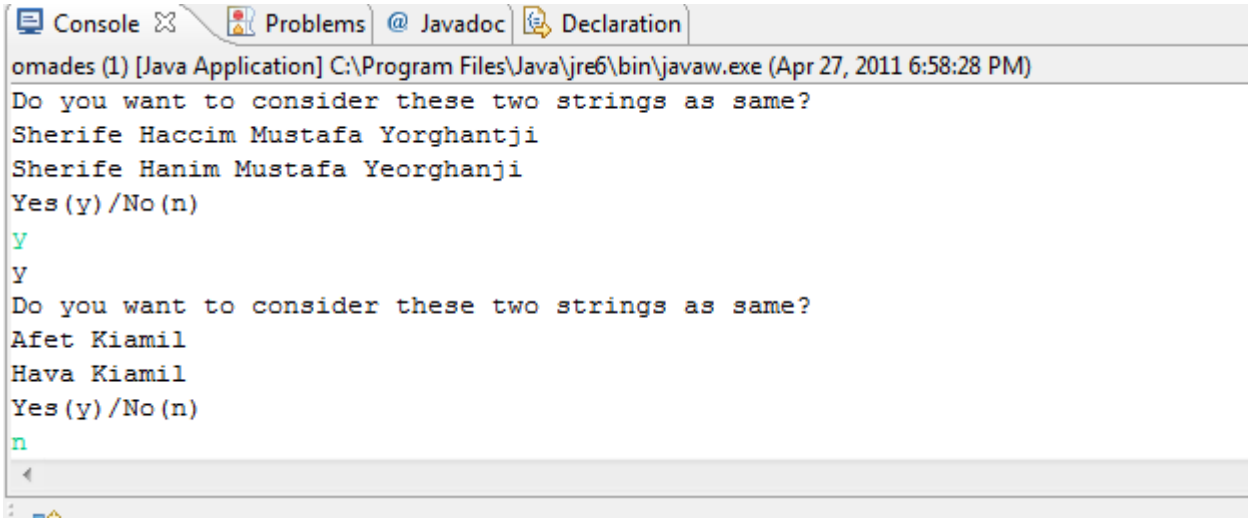
C h a r i s

H a r i s

Βλέπουμε ότι στην προκειμένη περίπτωση η αρχική μας προσέγγιση δεν θα δουλέψει γιατί δεν θα βρει κανένα κοινό χαρακτήρα και θα θεωρήσει ότι αυτά τα δυο string είναι τελείως διαφορετικά.

Μετά από αυτό παρατηρήσαμε ότι το πρόβλημα αυτό δεν ήταν εύκολο να λυθεί και έτσι ψάξαμε στο διαδίκτυο για πιθανές λύσεις και βρήκαμε μια βιβλιοθήκη η οποία είχε ενσωματωμένη μια μέθοδο η οποία βρίσκει πόσους χαρακτήρες έχουν διαφορά δυο strings.

Αφού βρήκαμε αυτή την βιβλιοθήκη, κάθε φορά που ελέγχουμε δυο strings να δούμε εάν είναι τα ίδια, σε περίπτωση που δεν είναι, τότε θα καλέσουμε την συγκεκριμένη μέθοδο για να δούμε πόσους χαρακτήρες έχουν διαφορά. Σε περίπτωση που η διαφορά τους είναι μικρότερη από 3 τότε θεωρούμε τα δυο string τα ίδια. Σε περίπτωση που έχουν διαφορά μικρότερη από 4 χαρακτήρες τότε ο χρήστης ρωτάται από το πρόγραμμα για να αποφασίσει εάν θέλει να θεωρηθούν τα δυο strings τα ίδια. Είναι πολύ εύκολο ο χρήστης να ρυθμίσει ο ίδιος στους πόσους χαρακτήρες θέλει να γίνεται η ερώτηση. Παρακάτω φαίνονται δυο παραδείγματα όπου ο χρήστης ρωτάται από το πρόγραμμα.




Στην πιο πάνω εικόνα παρατηρούμε επίσης γιατί είναι σημαντικό να ρωτήσουμε πρώτα τον χρήστη εάν θέλει να θεωρήσει δυο οποιαδήποτε strings, τα ίδια. Στην πρώτη περίπτωση βλέπουμε ότι ο χρήστης διαλέγει την επιλογή yes. Με αυτό δηλώνει ότι θέλει να θεωρήσει τα συγκεκριμένα strings σαν ένα και αυτό είναι λογικό και εμφανές. Στην δεύτερη περίπτωση όμως ο χρήστης επιλέγει την επιλογή no, κάτι που είναι επίσης εμφανές από ένα άνθρωπο αλλά όχι από υπολογιστή.

#### 4.3.2.2 Πρόβλημα 2

Το πιο πάνω πρόβλημα δεν ήταν το μόνο που αφορούσε το πώς ήταν γραμμένα τα όνομα κάποιων ιδιοκτητών και η πιο πάνω λύση δεν ήταν αρκετή. Το αρχείο γενικά δεν είχε συνοχή. Σε πολλές περιπτώσεις, επιπρόσθετα από το όνομα κάποιου ιδιοκτήτη αναγραφόταν και μια άλλη πληροφορία όπως, ποιού σύζυγος ήταν, ποιού γιος ήταν κλπ. Αυτό ήταν πρόβλημα γιατί ο ίδιος ιδιοκτήτης, όταν αναγραφόταν σαν γείτονας σε άλλες ιδιοκτησίες δεν αναγραφόταν αυτή η πληροφορία. Επίσης, υπήρχαν και άλλα προβλήματα, που ήταν πιο εύκολα να αντιμετωπιστούν, τα οποία είχαν να κάνουν με θέματα συνοχής, τα οποία λύθηκαν. Πιο κάτω φαίνεται ένα παράδειγμα.

2 16 121 Maria Athanasi wife of Costi Soteriou Parperi "street, Kallou Stavrino & Church of Chrysaliniotissa" "house of two rooms, one verandah, kitchen, yard & tree = [1100 sq ft]"

3 16 122 Kallou Stavrinis wife of Athanasi Kassabi "street, Styliani Avraami, Church of Chrysaliniotissa & Maria Athanasi" house of two rooms & right of passage through plot 121 = [430 sq ft]

Σε αυτό το παράδειγμα βλέπουμε ότι η ιδιοκτήτρια της ιδιοκτησίας 2, είναι η Maria Athanasi. Όμως βλέπουμε ότι αναγράφεται και μια πληροφορία για αυτήν(wife of Costi Soteriou Parperi). Το πρόβλημα είναι ότι στην ιδιοκτησία 3, όταν παρατηρήσουμε στους γείτονές της, βλέπουμε την Maria Athanasi όμως δεν αναγράφεται το wife of Costi Soteriou Parperi. Η λύση που βρήκαμε στο προηγούμενο πρόβλημα που είδαμε, βλέπουμε ότι δεν μπορεί να λύσει και αυτό το πρόβλημα γιατί τα δυο string(Maria Athanasi wife of Costi Soteriou Parperi και Maria Athanasi) δεν έχουν διαφορά 3 ή 4 χαρακτήρες, αλλά πολλούς περισσότερους. Επίσης ένα άλλο πρόβλημα που φαίνεται στο πιο πάνω παράδειγμα αφορά το πώς αναγράφονται οι γείτονες της κάθε ιδιοκτησίας. Τις περισσότερες φορές οι γείτονες της κάθε ιδιοκτησίας χωρίζονται με τον χαρακτήρα «,». Πολλές φορές όμως χωρίζονται και με τους χαρακτήρες «&» ή «/».
Η λύση για το πρόβλημα αυτό ήταν να κάνουμε κάποιες αλλαγές στον πίνακα. Δηλαδή, να αφαιρέσουμε όποιες περισσότερες πληροφορίες αναγράφονται για κάθε ιδιοκτήτη, να αντικαταστάσουμε όλα τα σύμβολα (&,/) με τον χαρακτήρα «,», έτσι ώστε να έχει το αρχείο καλύτερη συνοχή.

Υπήρχαν αρκετές όμως περιπτώσεις στις οποίες δεν ήταν αρκετή η λύση. Σε πολλές περιπτώσεις οι περισσότερες πληροφορίες που αναγράφονταν, δεν είχαν κάποια λέξη για να ξεχωρίσει το πρόγραμμα ότι υπήρχαν περισσότερες πληροφορίες όπως η λέξη wife. Το επόμενο παράδειγμα αναφέρεται σε αυτό το πρόβλημα.

43 16 162 Havuz Dervish Ezz ali Bey (school master) of Limassol "River, Kioulsoum ali Bey Halvaji and others ; street, Siddikka Hassan agha " "house of one upstairs room with verandah, one downstairs room with verandah, yard & trees= one evlek & 1488sqft"

Μετά από αυτή τν αλλαγή είχαμε ακόμη καλύτερα αποτελέσματα από τα αρχικά, αλλά σε σχέση με τα αποτελέσματα που θέλαμε να δούμε, υπήρχαν πολλές και μεγάλες διαφορές.

#### 4.3.2.3 Πρόβλημα 3

Όπως μπορούμε να παρατηρήσουμε και στα παραδείγματα των εγγραφών που έχουμε στο αρχείο πιο πάνω, για την κάθε ιδιοκτησία οι γείτονές της γράφονται με τα ονόματά τους και όχι με ένα μοναδικό αριθμό. Αυτό δημιούργησε μεγάλο πρόβλημα γιατί υπήρχαν πολλά άτομα τα οποία είχαν περισσότερες από μια ιδιοκτησίες στην περιοχή και αυτό προκαλούσε μεγάλη σύγχυση. Πιο κάτω βλέπουμε ένα παράδειγμα για την καλύτερη κατανόηση του προβλήματος από τον αναγνώστη.

24 16 142 Hazer Hussein "Hassan agha Hussein Karapartak, street, Government of Cyprus & Hassan agha Hussein Karapartak" "house of one room, one verandah, one kitchen with wc & yard = [ 2070 sq ft]"

Βλέπουμε ότι η ιδιοκτησία είναι του Hazer Hussein γειτονεύει με τον δρόμο, τον Hassan agha Hussein Karapartak και μια ιδιοκτησία που ανήκει στην κυβέρνηση της Κύπρου. Όμως η κυβέρνηση της Κύπρου έχει περισσότερες από μια ιδιοκτησίες σε εκείνη την περιοχή. Βλέπουμε πιο κάτω δυο παραδείγματα.

21 16 138 Government of Cyprus "Government of Cyprus,Municipality of Nicosia ,street & inside wall of Fortress" one store (old ammunition store)

25 16 143 Government of Cyprus "Hassan agha Hussein Karapartak; Haji Hussein of Mora, street" square = one evlek + 200 sqft

Αφού παρατηρήσαμε το πρόβλημα αυτό, σκεφτήκαμε ότι η ιδανικότερη λύση θα ήταν να διαβάσουμε τον δισδιάστατο πίνακα και για κάθε γείτονα της κάθε ιδιοκτησίας να τον αντιστοιχήσουμε με μια ιδιοκτησία. Δηλαδή αντί για κάθε γείτονα της κάθε ιδιοκτησίας, να αναγράφεται το όνομα του γείτονα, να αναγράφεται ένας μοναδικός αριθμός για τον συγκεκριμένο γείτονα και έτσι θα ξέρουμε με ποιους ακριβώς είναι γείτονας .Αυτός ο μοναδικός αριθμός θα είναι ο συνδυασμός του αριθμού σχεδίου(sheet) και ο αριθμός οικοπέδου.

Για να το κάνουμε αυτό σκεφτήκαμε να διαβάζουμε τον δισδιάστατο πίνακα και για κάθε γείτονα της κάθε ιδιοκτησίας, βλέπουμε σε πόσες ιδιοκτησίες είναι ιδιοκτήτης και κρατάμε ένα μετρητή. Σε περίπτωση που έχει μια ιδιοκτησία, παίρνουμε τον αριθμό σχεδίου και τον αριθμό οικοπέδου της ιδιοκτησίας εκείνης και αντικαθιστούμε αυτά με το όνομα του γείτονα. Στην περίπτωση όμως που ο συγκεκριμένος γείτονας είναι ιδιοκτήτης σε περισσότερες από μια ιδιοκτησίες, τότε πρέπει να ελέγξουμε για όλες αυτές τις ιδιοκτησίες, σε ποια από αυτές, ο ιδιοκτήτης (της ιδιοκτησίας που κοιτάζουμε να δούμε με ποιους γειτονεύει) αναγράφεται στους γείτονές της.
Έστω οι πιο κάτω εγγραφές στον πίνακα.

| | | | | |
|---|---|---|---|---|
| 1 | 16 | 120 | Α | Β,Γ |
| 2 | 16 | 121 | Α | Γ |
| 3 | 16 | 122 | Β | Α |
| 4 | 16 | 123 | Γ | Β |

Η Τρίτη ιδιοκτησία έχει τον Β σαν ιδιοκτήτη και γειτονεύει με τον Α. Ο Α όμως έχει δυο ιδιοκτησίες. Την 1 και την 2. Άρα πρέπει να κοιτάξουμε σε ποιες από τις δυο(1,2) αναγράφεται ο Β σαν γείτονάς της. Σε αυτή την περίπτωση είναι η πρώτη ιδιοκτησία.
Μετά από αυτή την αλλαγή στο πρόγραμμά μας, ο πιο πάνω πίνακας θα γίνει όπως φαίνεται πιο κάτω.

| | | | | |
|---|---|---|---|---|
| 1 | 16 | 120 | Α | 16 122,<br>16 123 |
| 2 | 16 | 121 | Α | 16 123 |
| 3 | 16 | 122 | Β | 16 120 |
| 4 | 16 | 123 | Γ | 16 122 |

#### 4.3.2.4 Πρόβλημα 4

Ακόμη ένα σοβαρό πρόβλημα που αντιμετωπίσαμε με το αρχείο είχε να κάνει με τις ιδιοκτησίες που ανήκαν σε περισσότερους από ένα ιδιοκτήτη. Το πρόβλημα με αυτές τες περιπτώσει ήταν ότι όταν στο αρχείο αναγραφόταν αυτή η ιδιοκτησία στους γείτονες μιας άλλης δεν αναγράφονταν όλοι οι ιδιοκτήτες της ιδιοκτησίας, αλλά μόνο ο ένας και σε πολλές περιπτώσεις αναγραφόταν με πολύ διαφορετικό τρόπο. Πιο κάτω φαίνεται ένα παράδειγμα εγγραφής μιας ιδιοκτησίας που έχει περισσότερους από ένα ιδιοκτήτες.

103 22 42 "Yorgho Tofalli,Erini Christodoulo" Xenou Yannaco wife of Panayioti Nicola; Zoi Hj Vassili Pervolari & street & Xenou Yannaco wife of Panayioti Nicola House of 2 rooms; kitchen with wc & yard = 763sqft

Πιο κάτω φαίνεται μια εγγραφή μιας ιδιοκτησίας όπου αναγράφει την πιο πάνω ιδιοκτησία στους γείτονές της.

104 22 43 Xenou Yannaco wife of Panayioti Nicola chrystallou Liassi Kouzari; Zoi Hj Vassili Pervolari; Yorghos Tofalli with his wife Erini Christodoulo & street & Chrystallou Liassi Kouzari "House of 3 downstairs rooms; verandah, kitchen, wc, one upstairs room, verandah, well = 1417sqft & trees"

Βλέπουμε ότι στην ιδιοκτησία 103, έχουμε δυο ιδιοκτήτες. Στην ιδιοκτησία 104 όπου αναγράφεται η 103 σαν γείτονάς της, βλέπουμε ότι υπάρχει πρόβλημα γιατί το πρόγραμμα δεν μπορεί να εντοπίσει με αυτό τον τρόπο την ιδιοκτησία 103.

Για αυτό το πρόβλημα σκεφτήκαμε ότι, κάθε φορά που κάνουμε την διαδικασία με τους γείτονες της κάθε ιδιοκτησίας εάν σε κάποιο γείτονα περιέχεται η λέξη with, τότε, όταν ψάχνουμε να τον ταιριάξουμε με μια ιδιοκτησία θα ψάχνουμε μόνο για ιδιοκτησίες που έχουν περισσότερους από ένα ιδιοκτήτη και θα ψάχνουμε εάν ανάμεσα στους ιδιοκτήτες της ιδιοκτησίας αυτής υπάρχει ο συγκεκριμένος γείτονας.

Αυτό δούλεψε για αρκετές περιπτώσει, όμως όχι για όλες. Λόγος ήταν αυτή η ασυνέπεια του αρχείου που προαναφέραμε. Εάν παρατηρήσουμε στο πιο κάτω παράδειγμα θα δούμε ότι στην ιδιοκτησία 6, στους γείτονες αναγράφεται σαν γείτονας ο Joannis Kyriakides όμως ο συγκεκριμένος ιδιοκτήτης, έχει την ιδιοκτησία 5, μαζί με κάποιους άλλους. Το πρόβλημα εδώ είναι ότι δεν αναγράφεται κάποια λέξη όπως with ή and others για να καταλάβουμε ότι η ιδιοκτησία έχει περισσότερους από ένα ιδιοκτήτες.

5 16 124 "Joannis Kyriakides, Sofoclis HjChristofi, Christofis Jacovaki,Yeorgios Pastellides & Costis Karayeorghi as trustees of Chrysaliniotissa Church Nicosia" Styliani Avraami "garden of two rooms, one stable & verandah & wheel well & tank = [eight donums, one evlek & 1725 sqft]"

6 16 124/1 "Church of Chrysaliniotissa,Varnava Christofi of Smyrna" "Katina Anastasi,Joannis Kyriakides & garden of Church of Chrysaliniotissa & street" [ 2800 sqft]

Αυτό είναι ένα από τα προβλήματα που αντιμετωπίσαμε με το αρχείο που δεν βρέθηκε η λύση τους. Υπάρχουν αρκετά παραδείγματα με παρόμοιο πρόβλημα στο αρχείο και αυτό δυσκόλεψε πολύ στο να βρούμε σωστά αποτελέσματα.

#### 4.3.2.5 Πρόβλημα 5

Στα πιο πάνω προβλήματα που αναφέραμε, βρήκαμε λύσεις(εκτός από το τελευταίο που είδαμε) και με κάθε πρόβλημα που λύναμε τα αποτελέσματα ήταν όλο και πιο σωστά όμως σε κάθε περίπτωση είχαν μεγάλη διαφορά από τα επιθυμητά αποτελέσματα. Αυτό οφείλεται σε πολλά προβλήματα που υπάρχουν στο αρχείο και αφορούν την συνέπεια και την πληρότητα του αρχείου. Λύνοντας κάποια πολύ πιο σοβαρά προβλήματα που δυστυχώς κάνουν το αρχείο να είναι ένα αναξιόπιστο δεδομένο εισόδου.

Κατά αρχήν, Για αρκετές ιδιοκτησίες, δεν υπάρχουν πληροφορίες για τους γείτονές τους, τους ιδιοκτήτες τους και αριθμούς οικοπέδων και σχεδίου. Αυτό λογικά θα

οφείλεται στην ηλικία των εγγράφων. Για αυτές τις ιδιοκτησίες το πρόγραμμά μας, δεν μπορεί να βρει την ομάδα στην οποία ανήκουν, ως αποτέλεσμα να υπάρχουν πολλές ιδιοκτησίες που δεν έχουν κάποια ομάδα. Πιο κάτω φαίνονται δυο τέτοια παραδείγματα.

182 22 120 " Sittica Mulla ali Yeorghanji, Halil Karir, Sittica Mulla Ali , Mehmed Halil Abdulbakir, Ahmed Halil Abdulbakir , Mustafa Halil Abdulbakir, Hussein Halil Abdulbakir " " " [ 2800 sqft]

Σε αυτό το παράδειγμα βλέπουμε ότι δεν έχουμε στοιχεία για τους γείτονες της ιδιοκτησίας.

88 Ismet Hannin Hussein wife of Suleiman Buyiaz Mehmed "street, Hassan agha Hussein Karapardak; Ismet Hannim Hussein wife of Suleiman Beyiaz Mehmed, Hassan Karapardak Hussein & Zia ali Abdullah Aschi and others" "House one room, two verandahs ; yard & tank & trees = 2295sqft"

Σε αυτό το παράδειγμα βλέπουμε ότι δεν έχουμε πληροφορίες για τον αριθμό σχεδίου το οποίο βρίσκεται και τον αριθμό οικοπέδου.

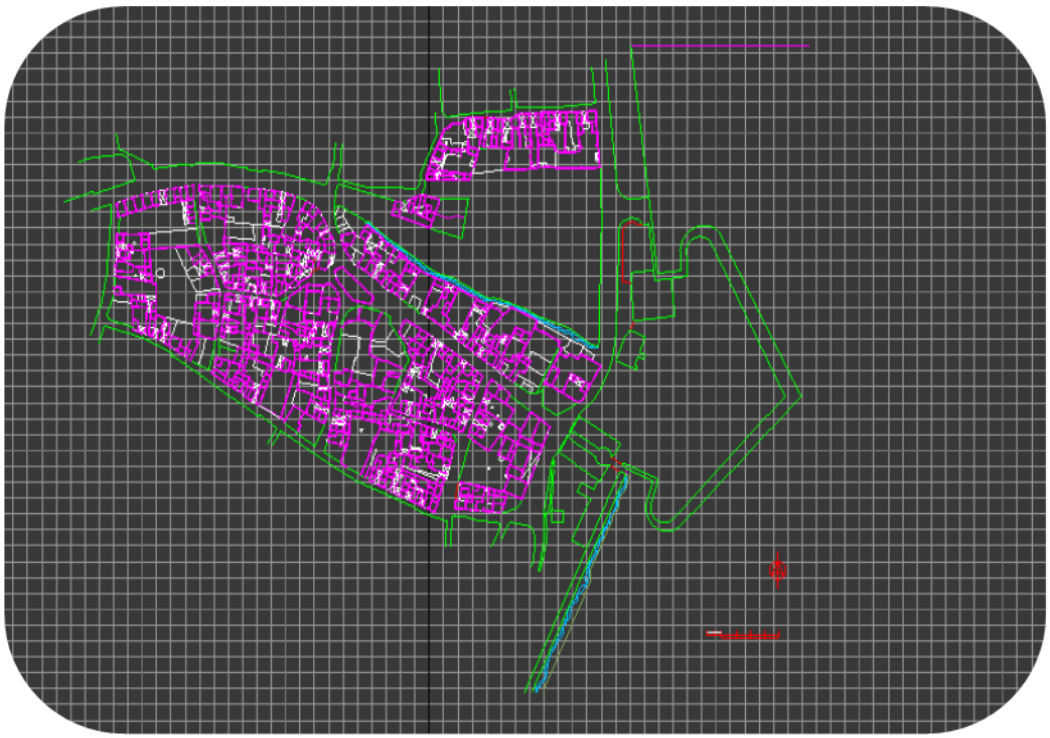
Άλλο τέτοιο πρόβλημα είναι ότι υπάρχουν πολλά λάθη στο αρχείο και το πρόγραμμα δεν μπορεί να τα αντιμετωπίσει αλλά ούτε και μπορούμε να βρούμε κάποια λύση. Ο λόγος για αυτά τα προβλήματα πιθανό να είναι ότι τα έγγραφα με τους τίτλους ιδιοκτησίας είναι πολύ παλιά, το ότι την εποχή εκείνη ο περισσότερος κόσμος δεν ήταν μορφωμένος και σε πολλές περιπτώσεις στο αρχείο αναγράφεται η φράση «pending valuations» και σημαίνει ότι αναμένονται αποτιμήσεις για την συγκεκριμένη ιδιοκτησία.

Στη συνέχεια, αφότου παρατηρήσαμε αυτά τα προβλήματα αποφασίσαμε να σταματήσουμε την προσπάθεια να βρούμε τις ομάδες γειτόνων με αυτό τον τρόπο με το συγκεκριμένο αρχείο, και προσπαθήσαμε να βρούμε κάποιο άλλο δεδομένο εισόδου, πιο αξιόπιστο με το οποίο να μπορέσουμε να βρούμε σωστά αποτελέσματα.

Το επόμενο πράγμα που σκεφτήκαμε ήταν με κάποιο τρόπο να βρούμε τον χάρτη της περιοχής, σε ηλεκτρονική μορφή, στον οποίο να φαίνονται τα όρια ιδιοκτησιών, για να μπορέσουμε να βρούμε τις ομάδες γειτόνων χρησιμοποιώντας τις κορυφές των ορίων των ιδιοκτησιών. Στο επόμενο υποκεφάλαιο θα αναλύσουμε την δεύτερη μεθοδολογία της εργασίας.

### 4.4 Μεθοδολογία 2

Η δεύτερη μεθοδολογία χρειάστηκε να γίνει λόγο των προβλημάτων που είδαμε προηγουμένως στο αρχείο που είχαμε σαν δεδομένο εσόδου. Μετά από τα πιο πάνω προβλήματα, ψάξαμε να βρούμε τον χάρτη της περιοχής σε κάποια ηλεκτρονική μορφή στον οποίο να φαίνονται τα όρια των ιδιοκτησιών και απευθυνθήκαμε στην Εσπερία όπου μας έδωσε ένα .dwg(drawing) αρχείο όπου σε αυτό υπήρχαν περίπου ένδεκα σχέδια και το ανοίξαμε με το λογισμικό AutoCAD. Το καθ' ένα περιείχε διαφορετικές πληροφορίες για τις ιδιοκτησίες της περιοχής. Αυτό που μας ενδιέφερε φυσικά εμάς, ήταν τα όρια των ιδιοκτησιών. Ευτυχώς, σε ένα από αυτά τα ένδεκα σχέδια υπήρχαν οι πληροφορίες για τα όρια ιδιοκτησιών. Στην εικόνα πιο κάτω φαίνεται το σχέδιο της περιοχής.



σχήμα 4.2: Χάρτης περιοχής απο το AutoCAD

Αφού βρήκαμε αυτή την πληροφορία σε ηλεκτρονική μορφή, σκεφτήκαμε να εξάγουμε κατά κάποιο τρόπο από το αρχείο, τις κορυφές των ορίων ιδιοκτησιών σε συντεταγμένες χ-ψ. Με αυτό τον τρόπο θα μπορούσαμε να βρούμε τους γείτονες με βάση τις κορυφές τους. Δηλαδή, εάν ξέραμε τις συντεταγμένες των κορυφών της κάθε ιδιοκτησίας, τότε με ένα απλό έλεγχο θα μπορούσαμε να αποφασίσουμε εάν δυο οποιεσδήποτε ιδιοκτησίες γειτονεύουν και με αυτό τον τρόπο θα μπορούσαμε να βρούμε τις ομάδες γειτόνων. Ο απλός έλεγχος θα μπορούσε να ήταν όπως πιο κάτω.

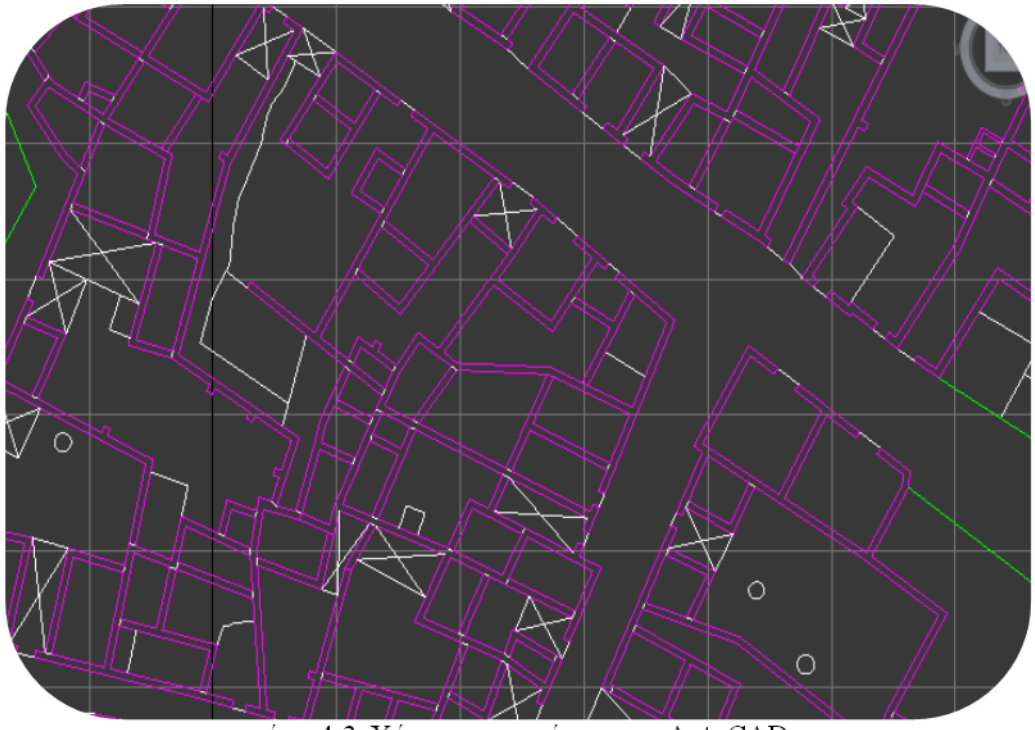
```
Foreach ownership A
        Foreach other ownership B
                If(A &B have at least 2 same vertices)
                        Means they are neighbors. Put B to neighbors of A;
```

Αφού εφαρμόσουμε αυτό τον αλγόριθμο, θα ξέρουμε τους γείτονες για την κάθε ιδιοκτησία, τότε θα μπορέσουμε να χρησιμοποιήσουμε τον αλγόριθμο που αναλύσαμε στην μεθοδολογία 1, για να υπολογίσουμε τις ομάδες γειτόνων.

Για να λειτουργήσει ο πιο πάνω αλγόριθμος όμως, πρέπει όταν εξάγουμε τις κορυφές, τα όρια των ιδιοκτησιών πρέπει να έχουν μια δομή για να μπορούμε να ξέρουμε ποια είναι τα τεμάχια των ιδιοκτησιών. Δηλαδή, εάν κάνουμε export τις κορυφές του σχεδίου στο σχήμα 4.4, χωρίς να έχουν καμία δομή, τότε θα έχουμε πολλές κορυφές χωρίς να έχουν κανένα νόημα και δεν θα μπορούσαμε να τις επεξεργαστούμε όπως είδαμε πιο πάνω.

Μετά από αρκετό ψάξιμο, κυρίως στο εγχειρίδιο χρήσης του λογισμικού AutoCAD , και χωρίς κανένα αποτέλεσμα, ανακαλύψαμε ότι αυτό μπορεί να γίνει στο λογισμικό 3D Studio Max. Φορτώσαμε το αρχείο στο συγκεκριμένο λογισμικό και εξάγαμε τις κορυφές των ορίων ιδιοκτησίας σε μορφή obj επιλέγοντας το layer oria idioktisiwn από το λογισμικό. Η δομή στην οποία αποθηκεύονται τα αρχεία obj, απλή και ευκολονόητη.

Αφού κάναμε export τις κορυφές, παρατηρήσαμε ότι υπήρχαν ιδιοκτησίες οι οποίες αποτελούνταν από περίπου 400 κορυφές και ο συνολικός αριθμός ιδιοκτησιών ήταν πολύ περισσότερος από αυτόν που έπρεπε να μας δώσει, κάτι που δεν ήταν καθόλου λογικό. Ο λόγος για αυτό το γεγονός ήταν ότι στο αρχείο τα όρια ιδιοκτησιών δεν ήταν καθόλου οργανωμένα και δεν είχαν σωστή δομή, δηλαδή δεν ήταν σωστά ορισμένα τα όρια ιδιοκτησιών. Ακόμη ένα πρόβλημα ήταν ότι οι γραμμές που όριζαν τα όρια ιδιοκτησιών ήταν διπλές και όχι μονές. Αυτό δημιουργούσε πρόβλημα γιατί οι γειτονικές ιδιοκτησίες δεν είχαν κοινές κορυφές. Στο επόμενο σχήμα βλέπουμε ένα παράδειγμα.



σχήμα 4.3: Χάρτης περιοχής απο το AutoCAD

Αφότου παρατηρήσαμε αυτά τα προβλήματα, αποφασίσαμε να κάνουμε export τα όρια των ιδιοκτησιών σαν .pdf αρχείο και στη συνέχεια να το μετατρέψουμε σε εικόνα, έτσι ώστε να υπολογίσουμε τις ομάδες γειτόνων χρησιμοποιώντας την εικόνα. Στο σχήμα 4.4 φαίνεται η εικόνα που πήραμε αφότου την εξάγαμε από το λογισμικό AutoCAD και την μετατρέψαμε από μορφή .pdf, σε μορφή .jpg

Σχήμα 4.4: Η εικόνα που εξάγαμε απο το AutoCAD

Παρατηρούμε ότι η εικόνα δεν είναι καλά συμπληρωμένη και υπάρχουν αρκετά κενά. Με αυτή την εικόνα δεν θα μπορούσαμε να επεξεργαστούμε τα όρια των ιδιοκτησιών γιατί εκτός του ότι δεν είναι συμπληρωμένες οι γραμμές, υπάρχουν και πολλές γραμμές οι οποίες δεν φαίνονται καν στο σχήμα και συνεπώς θα έχουμε και πάλι λανθασμένα αποτελέσματα.

Αφού δεν είχαμε άλλες επιλογές, σκεφτήκαμε να χρησιμοποιήσουμε το λογισμικό paint, για να συμπληρώσουμε τα κενά που υπάρχουν στην εικόνα. Στο σχήμα 4.4 φαίνεται η εικόνα του χάρτη αφού συμπληρώσαμε τα κενά με το λογισμικό paint.

Σχήμα 4.5: Η προηγούμενη εικόνα αφού την συμπληρώσαμε

Έχοντας λοιπών αυτή την εικόνα, ήμασταν σε θέση να την δώσουμε σαν είσοδο σαν είσοδο σε ένα πρόγραμμα το οποίο θα υπολογίζει τις κορυφές της κάθε ιδιοκτησίας και στη συνέχεια με βάση αυτές να υπολογίζει τους γείτονές της, καθώς επίσης και τις ομάδες γειτόνων της περιοχής.

Αυτό που κάναμε είναι να χρησιμοποιήσουμε την μέθοδο flood fill για να ξεχωρίσουμε την κάθε ιδιοκτησία και να υπολογίσουμε τις κορυφές της. Αυτό έπρεπε να γίνει γιατί, δεν μπορούσαμε απλά να υπολογίσουμε όλες τις κορυφές της εικόνας για το λόγο ότι θα είχαμε πληροφορίες μόνο για τις κορυφές της εικόνας χωρίς να ξέρουμε από ποιες κορυφές αποτελείται η κάθε ιδιοκτησία και αυτό δεν θα είχε κανένα νόημα να γίνει.

Η μέθοδος flood fill Παίρνει σαν είσοδο μια εικόνα, ένα εικονοστοιχείο(pixel) σε συντεταγμένες χ-ψ, ένα χρώμα στόχου(target color) και ένα χρώμα γεμίσματος(fill color) και ξεκινά από το pixel το οποίο παίρνει σαν είσοδο και προχωρά προς όλες τις κατευθύνσεις και γεμίζει όλα τα pixels με το χρώμα γεμίσματος μέχρι να φτάσει στο χρώμα στόχου.

Εμείς όμως δεν θέλαμε να κάνουμε ακριβώς αυτό που κάνει η μέθοδος, δηλαδή να γεμίσουμε όλα τα άσπρα pixels με κάποιο άλλο χρώμα μέχρις ότου να βρεθεί μια γραμμή, αλλά το αποτέλεσμα να το μεταφέρουμε σε μια καινούρια εικόνα, έτσι ώστε στη συνέχεια να επεξεργαστούμε στην εικόνα αυτή, όπου θα είναι μόνο μια ιδιοκτησία για να βρούμε τις κορυφές της.

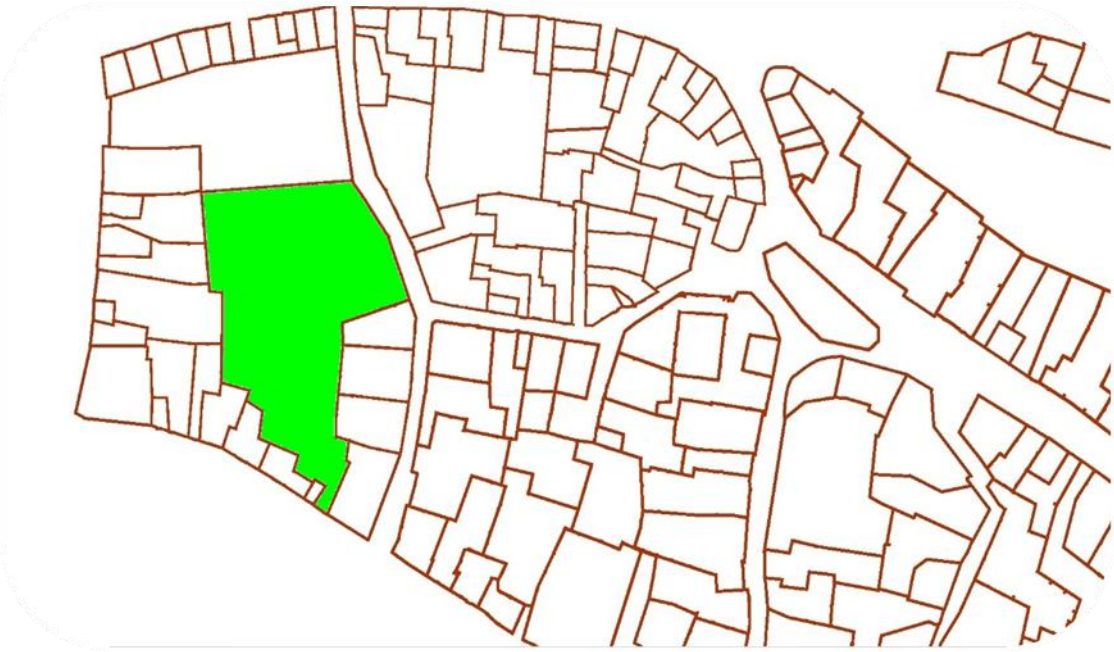
Στόχος μας ήταν να εφαρμόσουμε την μέθοδο αυτή για κάθε ιδιοκτησία και ακολούθως να υπολογίσουμε τις κορυφές της κάθε ιδιοκτησίας. Δηλαδή, για κάθε φορά (για κάθε ιδιοκτησία) που εφαρμόζουμε την μέθοδο να επεξεργαζόμαστε το κάθε αποτέλεσμα της μεθόδου σαν μια ξεχωριστή εικόνα. Με αυτό τον τρόπο μπορούμε να βρούμε σωστά τις κορυφές τις κάθε ιδιοκτησίας.

Στην αρχική μας προσπάθεια για να υλοποιήσουμε τον αλγόριθμο flood fill, χρησιμοποιήσαμε αναδρομή για να ελέγχουμε τα γειτονικά pixels και αυτό είχε ως αποτέλεσμα να έχουμε υπερφόρτωση(stack overflow) της στοίβας καλέσματος(call stack), λόγο των πολλών κλήσεων την συνάρτησης όπου υλοποιούσε τον αλγόριθμο. Ο λόγος τον οποίο είχαμε προβλήματα μνήμης είναι το ότι η εικόνα εισόδου είναι αρκετά μεγάλη ως αποτέλεσμα να χρειαστεί να ελεγχτούν πολλά pixels κάθε φορά. Η εικόνα εισόδου αποτελείται από 8,703,348 pixels (3508x2481).

Αυτό το πρόβλημα το αντιμετωπίσαμε με το να χρησιμοποιήσουμε μια στοίβα αντί αναδρομή. Με αυτό τον τρόπο δεν χρειάζεται να καλούμε πολλές συναρτήσεις αναδρομικά και δεν έχουμε προβλήματα μνήμης. Πιο κάτω φαίνεται ο αλγόριθμος που χρησιμοποιήσαμε στο πρόγραμμα.

```
void floodFill (Image img, int x, int y, int FillColor)
{
   Image NewImage=img;

   if(!push(x, y)) return;
   while(pop(x, y))
   {

     if(x + 1 < img->width)
                    color = getColor(img,x+1,y)
               if(isWhite(color))
          {
                              setColor(newImage,x+1,y,FillColor)
                              push(x+1,y)
          }
          if(x - 1 < 0)
                    color = getColor(img,x-1,y)
               if(isWhite(color))
          {
                              setColor(newImage,x-1,y,FillColor)
                              push(x-1,y)
          }
     if(y + 1 < img->height)
                    color = getColor(img,x,y+1)
               if(isWhite(color))
          {
                              setColor(newImage,x,y+1,FillColor)
                              push(x,y+1)
          }

      if(y - 1 < 0)
                    color = getColor(img,x,y-1)
               if(isWhite(color))
          {
                              setColor(newImage,x,y-1,FillColor)
                              push(x,y-1)
          }

   }

}
```

Ο πιο πάνω αλγόριθμος παίρνει σαν είσοδο μια εικόνα, τις συντεταγμένες ενός pixel(χ, ψ) και ένα χρώμα το οποίο είναι το χρώμα γεμίσματος και δημιουργεί μια νέα εικόνα με τις ίδιες διαστάσεις με την εικόνα εισόδου. Ακολούθως χρησιμοποιούμε μια στοίβα για να ελέγξουμε όλα τα pixels μέχρι να σταματήσουμε να βλέπουμε pixels με άσπρο χρώμα. Ο εξωτερικός βρόγχος, Κάνει pop στην στοίβα μέχρι να μην υπάρχουν άλλα άσπρα pixels και από τις τέσσερεις κατευθύνσεις(αριστερά, δεξιά, πάνω, κάτω). Σε κάθε επανάληψη του βρόγχου, ελέγχουμε για το pixel που κάναμε pop, τα τέσσερα γειτονικά του και αφού σιγουρευτούμε ότι είναι εντός των ορίων της εικόνας, ελέγχουμε εάν είναι άσπρο pixel,  δηλαδή δεν είναι το χρώμα της γραμμής του ορίου της συγκεκριμένης ιδιοκτησίας. Εάν ισχύει, τότε στην εικόνα γεμίζουμε με το χρώμα γεμίσματος, στο συγκεκριμένο pixel και το κάνουμε push την στοίβα για να ελέγξουμε και για αυτό τους γείτονές του στη συνέχεια. Αυτή η διαδικασία συνεχίζεται μέχρις ότου, να μην υπάρχουν άλλα pixels στην στοίβα. Στο τέλος αυτής της διαδικασίας, στην νέα εικόνα, έχουμε σκιαγραφημένα όλα τα pixels που αποτελούν την συγκεκριμένη ιδιοκτησία στην οποία ανήκει το pixel που δώσαμε στην flood fill σαν είσοδο στην αρχή. Στα σχήματα 4.6 και 4.7 βλέπουμε ένα παράδειγμα εφαρμογής του αλγορίθμου για μια ιδιοκτησία.



Σχήμα 4.6: Αποτέλεσμα αλγόριθμου FloodFill για μια ιδιοκτησία

Βλέπουμε ότι, με είσοδο ένα pixel το οποίο ήταν μέσα στην ιδιοκτησία που είναι σκιαγραφημένη με πράσινο χρώμα, αναγνωρίζουμε όλα τα pixels τα οποία αποτελείται η ιδιοκτησία και άρα μπορούμε να υπολογίσουμε όποια πληροφορία θέλουμε, για αυτή την ιδιοκτησία και μόνο. Στο επόμενο σχήμα βλέπουμε την καινούρια εικόνα που κρατάμε για την ιδιοκτησία και στη συνέχεια να την εξεργαστούμε και να βρούμε τις κορυφές της.

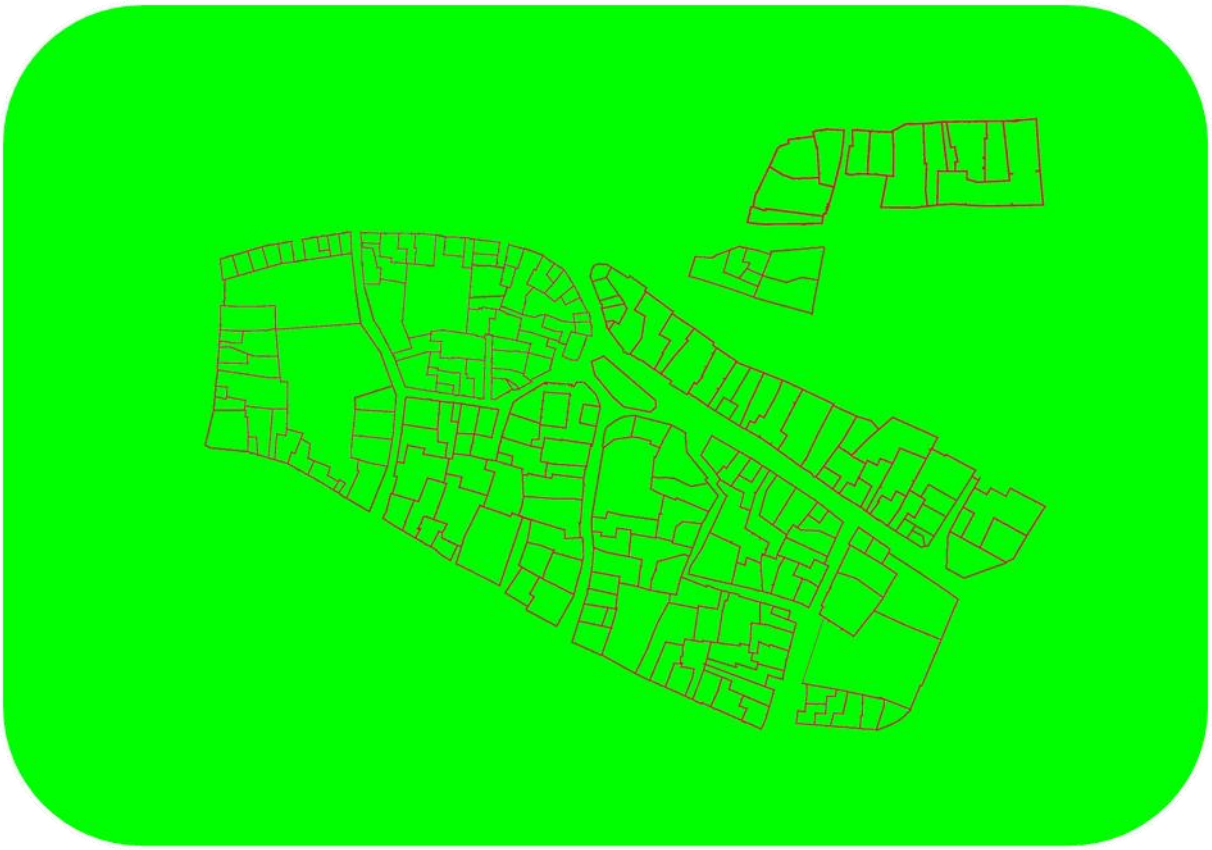
Σχήμα 4.7: Αποτέλεσμα αλγόριθμου FloodFill για μια ιδιοκτησία

Με αυτό τον τρόπο λοιπών βρήκαμε τα όρια μιας ιδιοκτησίας. Αυτό που θέλουμε όμως είναι να βρούμε τα όρια όλων των ιδιοκτησιών. Δηλαδή αυτό που κάναμε για την ιδιοκτησία που είδαμε στα πιο πάνω σχήματα, να το κάνουμε για όλες τις ιδιοκτησίες, έτσι ώστε για κάθε ιδιοκτησία ξεχωριστά, να υπολογίσουμε τις κορυφές της. Χρησιμοποιώντας λοιπών την πιο πάνω μέθοδο, προσθέτουμε ένα εξωτερικό βρόγχο και εφαρμόζουμε το flood fill για κάθε ιδιοκτησία. Πιο κάτω φαίνεται ο αλγόριθμος.

```
for(i=0; i<imageWidht; i++)
        for(j=0; j<imageHeight; j++)
                color=getColor(image,I,j);
                        if(isWhitr(color))
                                FloodFill(image,I,j);
```

Με τον πιο πάνω αλγόριθμο καλούμε για κάθε άσπρο pixel την συνάρτηση Flood fill και σε κάθε επανάληψη ελέγχουμε εάν το παρόν pixel είναι άσπρο. Δηλαδή, δεν είναι ούτε το χρώμα των ορίων αλλά ούτε και το χρώμα που γεμίζουμε την εικόνα. Κάνουμε αυτό τον έλεγχο για να είμαστε σίγουροι ότι δεν θα υπολογίσουμε περισσότερες από μια φορές μια ιδιοκτησία. Ακολούθως, για να σιγουρευτούμε ότι δεν χάνουμε κάποια ιδιοκτησία με αυτό τον τρόπο, αφού προσθέσαμε την πιο πάνω επανάληψη, τυπώσαμε

το αποτέλεσμα της εικόνας εισόδου, μετά από το τέλος της επανάληψης. Η εικόνα φαίνεται στο επόμενο σχήμα.



Σχήμα 4.8: Αποτέλεσμα αλγόριθμου Flood Fill για όλη την εικόνα εισόδου

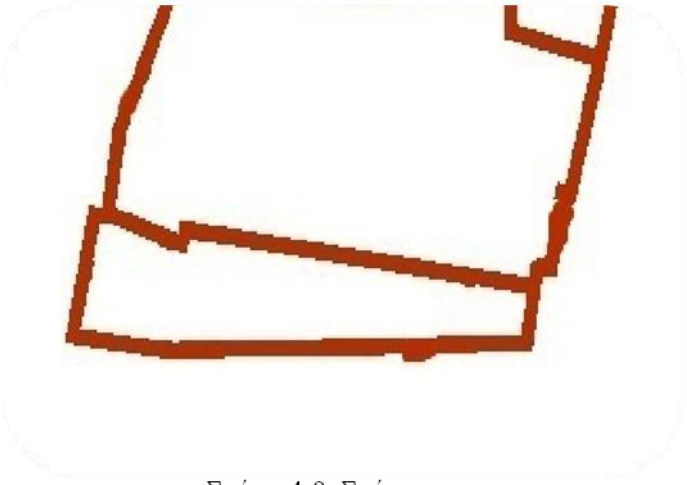
Αφού σιγουρευτήκαμε ότι ο αλγόριθμος λειτουργεί σωστά, γνωρίζουμε ότι έχουμε την κάθε ιδιοκτησία σαν μια ξεχωριστή εικόνα και μπορούμε να την επεξεργαστούμε διαφορετικά από τις υπόλοιπες. Αυτό που σκεφτήκαμε να κάνουμε ήταν να εφαρμόσουμε στην κάθε ιδιοκτησία, δηλαδή, στην κάθε εικόνα που μας δίνει ο αλγόριθμος flood fill σε κάθε επανάληψη του εξωτερικού βρόγχου, ανίχνευση γωνιών (corner detection).

Άρα για κάθε επανάληψη του εξωτερικού βρόγχου, αφού εφαρμόσουμε το Flood Fill, στη συνέχεια καλούμε μια συνάρτηση της opencv όπου με είσοδο μια εικόνα, επιστρέφει σαν έξοδο, τις γωνιές που βρίσκονται σε αυτή. Πιο κάτω φαίνεται η κλήση την συνάρτησης.

```
cvGoodFeaturesToTrack(GrayTemaxio,Copy_of_temaxio,Copy_of_t
emaxio_2,corners,&corners_count,0.07,15,NULL,9,0);
```

Οι παράμετροι εισόδου της συνάρτησης, είναι η εικόνα που θα εφαρμοστεί το corner detection, δυο αντίγραφα της εικόνας τα οποία πρέπει να έχουν το ίδιο μέγεθος με την εικόνα, ένας πίνακας από σημεία, όπου σε αυτά θα επιστραφούν τα σημεία των γωνιών, ένας μετρητής ο οποίος θα ισούται με τον αριθμό των γωνιών που εντοπίστηκαν, ένας αριθμός όπου με αυτόν δηλώνουμε πια μέθοδος ανίχνευσης θα χρησιμοποιηθεί (στη δική μας περίπτωση επιλέγηκε το Harris corner detection) και επίσης κάποιοι παράμετροι με τους οποίους ο χρήστης μπορεί να καθορίσει το τι είναι γωνιά. Χρειάστηκαν αρκετές προσπάθειες για να δώσουμε τις κατάλληλες παραμέτρους έτσι ώστε να παίρνουμε τα ιδανικά αποτελέσματα.

Με αυτό τον τρόπο υπήρχαν δυο προβλήματα. Η συνάρτηση όταν επέστρεφε τα σημεία των γωνιών δεν τα επέστρεφε με μια σωστή σειρά. Δηλαδή, για παράδειγμα εάν είχαμε μια ιδιοκτησία με πέντε γωνιές, τότε η συνάρτηση θα μας επέστρεφε τις πέντε αυτές γωνιές χωρίς να είναι ταξινομημένα αριστερόστροφα είτε δεξιόστροφα. Αυτό δημιουργούσε πρόβλημα γιατί εάν είχαμε απλά αυτά τα πέντε σημεία για αυτή την ιδιοκτησία, τότε, δεν μπορούμε να ξέρουμε το σχήμα της. Απλά θα ξέρουμε ότι οι γωνιές της είναι αυτά τα πέντε σημεία. Το άλλο πρόβλημα που είχαμε και δεν αφορούσε όμως τον συγκεκριμένο αλγόριθμο ήταν ότι οι γραμμές της εικόνας που χώριζαν τα όρια ιδιοκτησιών ήταν πολύ χοντρές. Αυτό είναι πρόβλημα γιατί όταν θα βρίσκαμε τις κορυφές της κάθε ιδιοκτησίας, μετά θα θέλαμε να τις συγκρίνουμε με τις κορυφές των άλλων ιδιοκτησιών και τότε δεν θα βρίσκαμε κοινές κορυφές για το λόγο ότι οι γραμμές ήταν πολύ χοντρές και οι γειτονικές ιδιοκτησίες στην ουσία θα βρίσκονταν μακριά η μια με την άλλη. Στο επόμενο σχήμα φαίνεται μια εικόνα όπου θα δούμε γιατί γίνεται το πιο πάνω φαινόμενο.



Σχήμα 4.9: Σχήμα απο
μεγένθηνσητης εικόνας εισόδου

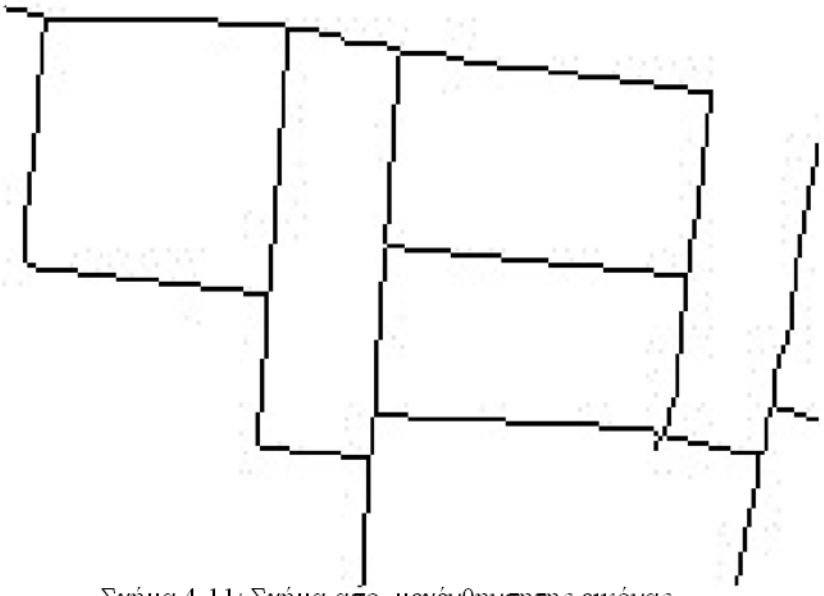
Παρατηρούμε ότι οι δυο πιο πάνω ιδιοκτησίες γειτονεύουν. Με την μέθοδο αυτή όμως δεν θα μπορέσουμε να βρούμε ότι γειτονεύουν, γιατί οι κορυφές των δυο ιδιοκτησιών απέχουν πολλά pixels η μια με την άλλη.

Για να λύσουμε αυτό το πρόβλημα, έπρεπε να χρησιμοποιήσουμε ένα αλγόριθμο ο οποίος λέγεται image thinning ή skeletonize. Ο αλγόριθμος αυτός παίρνει σαν είσοδο μια εικόνα και επιστρέφει την εικόνα, με όλες της τις γραμμές να έχουν πάχος μόνο ένα pixel. Αυτό είναι πολύ χρήσιμο γιατί τότε θα μπορούμε να υπολογίσουμε τις γειτονικές ιδιοκτησίες πολύ πιο εύκολα αφού θα βρίσκονται πολύ πιο κοντά σε απόσταση pixels.

Αφού αποφασίσαμε ότι έπρεπε να χρησιμοποιήσουμε αυτό τον αλγόριθμο, βρήκαμε ένα πρόγραμμα το οποίο έκανε ατή τη δουλειά και το χρησιμοποιήσαμε για να κάνουμε την εικόνα μας πιο λεπτή. Στο επόμενο σχήμα φαίνεται το αποτέλεσμα.

Σχήμα 4.10 Το αποτέλεσμα της εφαρμογής του αλγόριθου skeletonize

Βλέπουμε ότι οι γραμμές δεν είναι τόσο εμφανείς για το λόγο ότι έχουν πάχος μόλις ένα pixel. Στο επόμενο σχήμα θα δούμε την ίδια εικόνα σε μεγέθυνση.



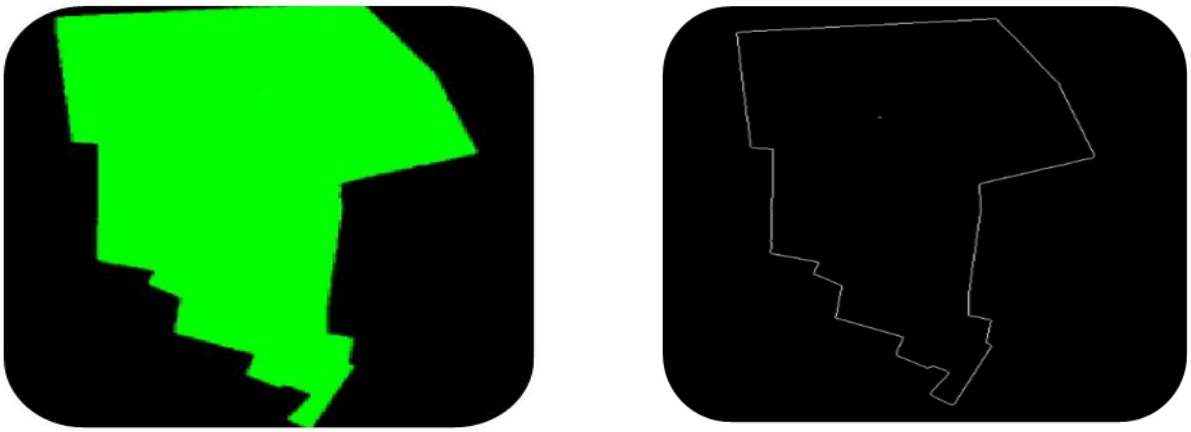
Σχήμα 4.11: Σχήμα απο μεγένθηνσητης εικόνας μετά την εφαρμογή του skeletonize

Αφού λύσαμε το πρώτο πρόβλημα που είχαμε, τώρα έπρεπε να βρούμε ένα τρόπο να έχουμε τα σημεία της κάθε ιδιοκτησίας με μια σωστή σειρά. Αφού μελετήσαμε πολλές συναρτήσεις της opencv, βρήκαμε μια συνάρτηση η οποία παίρνει σαν παράμετρο εισόδου μια εικόνα και μας επιστρέφει το περίγραμμά της. Η συνάρτηση η οποία κάνει αυτή τη δουλειά είναι η `cvFindContours`. Η κλήση της συνάρτησης φαίνεται πιο κάτω.

```
cvFindContours(GrayTemaxio,mem,&seq,size,CV_RETR_EXTERNAL,CV_LINK_RUNS
);
```

Η συνάρτηση παίρνει σαν παραμέτρους εισόδου μια εικόνα, μια μεταβλητή μνήμης τύπου CvMemStorage , μια μεταβλητή τύπου CvSeq, ένα ακαίρεο όπου συνήθως αρχικοποιείται με ένα σταθερό μέγεθος και οι δυο τελευτάιες παραμέτροι ορίζουν το πώς θέλουμε να έχουμε τα αποτελέσματα την συνάρτησης. Με τον τρόπο που δίνουμε τις δυο αυτές παραμέτρους, ορίζουμε ότι θέλουμε να έχουμε τα αποτελέσματα σε σημεία.

Αφού δώσαμε τις κατάλληλες παραμέτρους εισόδου, η συνάρτηση μας επέστρεψε τις σωστές συντεταγμένες για τα περιγράμματα. Αυτό το διαπιστώσαμε καλώντας τη συνάρτηση αυτή για μια ιδιοκτησία και ακολούθως χρησιμοποιόντας τη συνάρτηση cvDrawContours ζωγραφίσαμε τα περιγράμματα που μας επέστρεψε η συνάρτηση. Στο επόμενο σχήμα φαίνεται ένα παράδειγμα.



Σχήμα 4.12: Τα περιγράμματα όπως μας τα επιστρέφει η συνάρτηση CvFindContours

Αφού σιγουρευτήκαμε ότι τα αποτελέσματα που μας επιστρέφει η συνάρτηση είναι σωστά, το επόμενο βήμα είναι με βάση τα περιγράμματα της κάθε ιδιοκτησίας, να βρούμε για κάθε ιδιοκτησία του γείτονες της έτσι ώστε στη συνέχεια να βρούμε τις ομάδες γειτόνων. Αποφασίσαμε, να υλοποιήσουμε μια κλάση τύπου temaxio, έτσι ώστε για κάθε ιδιοκτησία να κρατούμε τα σημεία των περιγραμμάτων της. Σε κάθε επανάληψη λοιπών του εξωτερικού βρόγχου, δημιουργούμε ένα καινούριο αντικείμενο αποθηκεύουμε σε αυτό τα σημεία των περιγραμμάτων, και προσθέτουμε το αντικείμενο σε ένα πίνακα όπου είναι και αυτός τύπου temaxio. Όταν τελειώσει η διαδικασία αυτή, έχουμε ένα πίνακα με όλες τις ιδιοκτησίες μέσα και η για κάθε ιδιοκτησία κρατούμε τα σημεία που αποτελούν το περίγραμμά τους καθώς επίσης και από πόσα σημεία αποτελείται η ιδιοκτησία.

Αυτό που πρέπει να γίνει στη συνέχεια είναι να υπολογίσουμε για κάθε ιδιοκτησία όλους τους γείτονές της. Αυτό το υλοποιούμε με ένα βρόγχο όπου για κάθε ιδιοκτησία ελέγχουμε με την κάθε άλλη ιδιοκτησία που υπάρχει, εάν είναι γείτονες. Άρα για αυτό χρειαζόμαστε ακόμη ένα εσωτερικό βρόγχο ο οποίος θα περνά και αυτός από κάθε ιδιοκτησία. Άρα σε κάθε επανάληψη του εσωτερικού βρόγχου, ελέγχουμε εάν οι δυο ιδιοκτησίες(εσωτερικού και εξωτερικού βρόγχου) έχουν κοινά σημεία σαν περίγραμμα. Για τον λόγω του ότι δεν μπορούν να έχουν σημεία τα οποία να είναι ακριβώς τα ίδια, επειδή είναι χωρισμένα από μια γραμμή, ελέγχουμε εάν τα δυο σημεία χ και ψ, των δυο ιδιοκτησιών έχουν διαφορά μόνο 4 pixels. Αυτό όμως πρέπει να ισχύει και για το χ αλλά και για το ψ. Κάθε τέλος του εσωτερικού βρόγχου γράφουμε σε ένα αρχείο τους γείτονες της ιδιοκτησίας εκείνης. Πιο κάτω φαίνεται ο αλγόριθμος που χρησιμοποιούμε.

```
ForEachOwnership i
        ForEachOwnerShip j{
                If(i!=j)
                        If(ownership i is neighbor with ownership j)
                                Add to a table the ownership j
        }
        WhriteALLElementsOfTheTableToTheFile
```

Ο τρόπος με τον οποίο αποθηκεύουμε τους γείτονες στο αρχείο είναι ο εξής: Στην πρώτη στήλη αποθηκεύουμε τον αριθμό της ιδιοκτησίας και στη δεύτερη στήλη αποθηκεύουμε τους γείτονές της και τους χωρίζουμε με τον χαρακτήρα «,». Πιο κάτω φαίνονται οι πρώτες πέντε εγγραφές στο αρχείο.

1 2,13,14
2 1,3,4,13
3 2,4
4 2,3,5,13,17,19
5 4,6,7,19

Το αρχείο αυτό αποτελείται από 227 ιδιοκτησίες, όσες ακριβώς υπάρχουν και στην εικόνα εισόδου.

Αφού λοιπόν έχουμε αυτό το αρχείο με τις όλες τις ιδιοκτησίες της περιοχής και τους γείτονές τους, τότε είμαστε σε θέση να βρούμε τις ομάδες γειτόνων της περιοχής γιατί τώρα έχουμε ένα πιο αξιόπιστο και ορθό δεδομένο εισόδου. Αυτό που σκεφτήκαμε να κάνουμε είναι να δώσουμε αυτό το αρχείο σαν είσοδο στο πρόγραμμα που αναπτύξαμε στην γλώσσα προγραμματισμού java, το οποίο βρίσκει με είσοδο ένα παρόμοιο αρχείο με αυτό, τις ομάδες γειτόνων.

Το πρόγραμμα που αναπτύξαμε όμως για να δουλέψει με αυτό το αρχείο σαν δεδομένο εισόδου, θα πρέπει να υποστεί κάποιες αλλαγές. Όταν κάναμε τις απαραίτητες αλλαγές στο πρόγραμμα, τότε πήραμε σαν έξοδο τις ομάδες γειτόνων της περιοχής.

Για να ελέγξουμε εάν τα αποτελέσματα είναι σωστά, μετρήσαμε πόσες ομάδες γειτόνων έχει βρει και ο αριθμός ήταν δώδεκα, όσο και ο αριθμός των ομάδων που βρίσκονται και στην εικόνα εισόδου. Αυτό όμως δεν ήταν αρκετό για να σιγουρευτούμε ότι τα αποτελέσματα είναι σωστά και στη συνέχεια μετρήσαμε για την κάθε ομάδα στην εικόνα από πόσες ιδιοκτησίες αποτελείται και συνέχεια κάναμε το ίδιο με τα αποτελέσματα που μας έχει δώσει το πρόγραμμα. Και σε αυτή την περίπτωση επίσης οι αριθμοί ήταν ακριβώς οι ίδιοι. Επίσης, αφού ξέραμε πιο τεμάχιο στην εικόνα ήταν η

πρώτη εγγραφή στο αρχείο, κάναμε ένα έλεγχο με ποιους γειτονεύει στην εικόνα και με ποιους γειτονεύει στα αποτελέσματα του αρχείου και η σύγκριση μας έδειχνε ότι είχαμε τα σωστά αποτελέσματα.

# Κεφάλαιο 5

## Αποτελέσματα

---

5.1 Γενικά .....60
5.2 Αποτελέσματα .....60
5.2 Συμπεράσματα .....63
5.2.1 Μελλοντική Δουλειά .....63
5.2.2 Εμπειρίες που Αποκτήθηκαν .....65

---

### 5.1 Γενικά

Σε αυτό το κεφάλαιο παραθέτουμε συνοπτικά τα διάφορα στάδια της υλοποίησης της εργασίας, τα αποτελέσματα της εργασίας, τα συμπεράσματα και επίσης θα αναφερθούμε στην προοπτική της μελλοντικής δουλειάς.

### 5.2 Αποτελέσματα

Ο στόχος της διπλωματικής μου εργασίας ήταν η τρισδιάστατη μοντελοποίηση της παλιάς Λευκωσίας του 19$^{ου}$ αιώνα με αυτοματοποιημένο τρόπο. Αυτό όμως δεν επιτεύχθηκε για το λόγο ότι συναντήσαμε κατά καιρούς πολλά προβλήματα με τα αρχεία και τα δεδομένα εισόδου. Αυτά τα προβλήματα που βρίσκαμε μας καθυστερούσαν πολύ και σε πολλές περιπτώσεις μας έκαναν να αλλάξουμε κατεύθυνση για να πετύχουμε τον στόχο μας.

Ξεκινήσαμε λοιπόν με το να μετατρέψουμε το αρχείο κειμένου το οποίο περιείχε τους τίτλους ιδιοκτησίας για την συγκεκριμένη περιοχή της παλιάς Λευκωσίας, σε μορφή csv για να μπορέσουμε να διαβάσουμε το αρχείο από το πρόγραμμά μας με ένα εύκολο και αποδοτικό τρόπο. Στη συνέχεια, έπρεπε με βάση αυτό το αρχείο να υπολογίσουμε

τις ομάδες γειτόνων και τα όρια των ιδιοκτησιών για να τα περάσουμε σε μεταγενέστερο στάδιο και να μοντελοποιήσουμε την πόλη.

Όπως αναφέραμε όμως και σε προηγούμενα κεφάλαια, στο αρχείο συναντούσαμε προβλήματα κατά τη διάρκεια της υλοποίησης του προγράμματος. Κάποια από αυτά όπως είδαμε λύθηκαν και κάποια άλλα όχι λόγο ασυνέπειας που υπήρχε στο αρχείο.

Αυτή η ασυνέπεια λογικά οφείλεται στην ηλικία των τίτλων ιδιοκτησίας και στο ότι πολλοί άνθρωποι την εποχή εκείνη δεν είχαν αρκετή μόρφωση και έτσι μπορούμε να παρατηρήσουμε ιδιοκτήτες να γράφονται διαφορετικά στο αρχείο, και πολλά άλλα προβλήματα. Αφού υλοποιήσαμε τον αλγόριθμο για την εύρεση των γειτόνων δεν μπορούσαμε να δούμε σωστά αποτελέσματα για το λόγο που προαναφέραμε. Τα αποτελέσματα που πήραμε από το πρόγραμμα ήταν περίπου 55-65% σωστά σε σχέση με τα αποτελέσματα που βρήκε η Εσπερία στην δική της εργασία, αλλά όπως αναφέραμε και στο κεφάλαιο 2, η Εσπερία είχε υπολογίσει αυτές τις πληροφορίες, χειροκίνητα και όχι με ανάπτυξη κάποιου προγράμματος, και μπορούσε να αντιληφτεί κάποια στοιχεία που μπορούσε να ήταν λογικά, αλλά ένα πρόγραμμα δεν μπορεί να τα δει.

Μετά από αυτό, αποφασίσαμε με τον καθηγητή μου κ. Χρυσάνθου, ότι με το δεδομένο αρχείο δεν θα μπορέσουμε να καταλήξουμε στα επιθυμητά αποτελέσματα και ότι θα πρέπει να βρούμε ένα άλλο τρόπο για να ανακτήσουμε τις πληροφορίες που θέλουμε για να προχωρήσουμε στο επόμενο βήμα. Όταν απευθυνθήκαμε στην Εσπερία για αυτά τα προβλήματα, μας είχε δώσει τον χάρτη της περιοχής σε μια ηλεκτρονική μορφή (drawing). Αυτό, θα μας βοηθούσε έτσι ώστε να εξάγουμε τις κορυφές των ιδιοκτησιών και να υπολογίσουμε τις ομάδες γειτόνων και τα όρια ιδιοκτησιών. Σαν αποτέλεσμα αυτής της διαδικασίας, είχαμε ένα αρχείο obj με τις κορυφές των ιδιοκτησιών. Και σε αυτό το αρχείο όμως συναντήσαμε πρόβλημα γιατί οι ιδιοκτησίες δεν ήταν καλά οριοθετημένες και τα σημεία των κορυφών δεν μπορούσαν να μας βοηθήσουν γιατί, ήταν απλά κάποια σημεία χωρίς να έχουν κάποιο νόημα.

Αυτό που σκεφτήκαμε να κάνουμε στη συνέχεια ήταν να εξάγουμε τον χάρτη από το λογισμικό AutoCAD και να υπολογίσουμε τις κορυφές των ιδιοκτησιών

επεξεργάζοντας την εικόνα. Αυτό που θέλαμε σε αυτή την φάση ήταν να υπολογίσουμε τα περιγράμματα της κάθε ιδιοκτησίας για να υπολογίσουμε στην συνέχεια τις πληροφορίες που θέσαμε ως στόχο εξ αρχής. Για να μπορέσουμε να επεξεργαστούμε την κάθε ιδιοκτησία ξεχωριστά, αποφασίσαμε να εφαρμόσουμε τον αλγόριθμο flood fill. Αφού υλοποιήσαμε τον αλγόριθμο παρατηρήσαμε ότι για το λόγω του ότι η εικόνα ήταν μεγάλου μεγέθους, οι γραμμές των ορίων ιδιοκτησίας, είχαν πάχος πολλά pixels, κάτι που έκανε τον υπολογισμό των γειτόνων πολύ δύσκολο. Για αυτό το λόγο σκεφτήκαμε να εφαρμόσουμε στην εικόνα μας αυτό που λέγεται image thinning για να κάνουμε τις γραμμές των ιδιοκτησιών πιο λεπτές.

Αφού πετύχαμε στην εικόνα μας το image thinning, η συσχέτιση των ιδιοκτησιών ήταν πολύ πιο εύκολη από πριν. Μετά από αυτό χρησιμοποιήσαμε το corner detection για να υπολογίσουμε τις κορυφές της κάθε ιδιοκτησίας και να υπολογίσουμε τους γείτονες με αυτό τον τρόπο. Αφού εφαρμόσαμε το corner detection, είχαμε τις κορυφές της κάθε ιδιοκτησίας, όμως το πρόβλημα με αυτή τη μεθοδολογία ήταν ότι οι κορυφές που μας επέστρεφε η μέθοδος δεν ήταν ταξινομημένες με κάποια σειρά, δεξιόστροφη ή αριστερόστροφη.

Στη συνέχεια χρησιμοποιήσαμε μια συνάρτηση της opencv για να βρούμε τα περιγράμματα της κάθε ιδιοκτησίας και σε αυτή τη περίπτωση τα σημεία μας είχαν σωστή σειρά και έτσι μπορούσαμε να προχωρήσουμε στο επόμενο βήμα.

Αφού πλέον είχαμε τα περιγράμματα της κάθε ιδιοκτησίας ξεχωριστά, μπορούσαμε να υπολογίσουμε τους γείτονες της κάθε ιδιοκτησίας και έτσι έγινε. Αποθηκεύουμε τους γείτονες της κάθε ιδιοκτησίας σε ένα txt αρχείο και στη συνέχεια δίνουμε αυτό το αρχείο στο πρόγραμμα που υλοποιήσαμε προηγουμένως και με λίγες αλλαγές στο πρόγραμμα πετύχαμε την εύρεση των ομάδων γειτόνων. Το πρόγραμμα βρίσκει 227 ιδιοκτησίες(όσες και στην εικόνα) και 12 ομάδες γειτόνων όπως επίσης και στην εικόνα.

Σχήμα 5.1: Εικόνα εισόδου του προγράμματος

Παρατηρούμε ότι στην εικόνα 5.1 υπάρχουν 12 ομάδες γειτόνων, όσες και ο αριθμός που βρίσκει το πρόγραμμα. Χρησιμοποιήσαμε πολλούς τρόπους για να επαληθεύσουμε τα αποτελέσματα του προγράμματος όπως, να μετρήσουμε τον αριθμό ιδιοκτησιών που υπάρχουν στην κάθε περιοχή. Επίσης αφού γνωρίζουμε ποια είναι η πρώτη ιδιοκτησία, δηλαδή η ιδιοκτησία στο αρχείο μας με το νούμερο 1, κάναμε μια σύγκριση μεταξύ τους γείτονές της που φαίνονται στην εικόνα και αυτούς που αναγράφονται στο αρχείο των γειτόνων της κάθε ιδιοκτησίας. Με αυτό τον τρόπο ελέγξαμε και άλλες, αρκετές ιδιοκτησίες για να βεβαιωθούμε ότι τα αποτελέσματα μας ήταν σωστά. Τα αποτελέσματα με τις ομάδες γειτόνων τα αποθηκεύουμε επίσης σε ένα txt αρχείο. Επίσης είναι σημαντικό να αναφερθεί ότι εκτός από τις ομάδες γειτόνων έχουμε και τα περιγράμματα της κάθε ιδιοκτησίας σε συντεταγμένες. Με αυτό τον τρόπο μπορούμε να γνωρίζουμε την ακριβές τοποθεσία της κάθε ιδιοκτησίας.

## 5.3 Συμπεράσματα

### 5.3.1 Μελλοντική Δουλειά

Όπως σε κάθε εργασία, έτσι και σε αυτή υπάρχουν προοπτικές για περεταίρω δουλειά στην εργασία. Όπως προαναφέραμε, για τα πολλά προβλήματα που συναντήσαμε, δεν καταφέραμε να φτάσουμε στο σημείο όπου θα μοντελοποιούσαμε την πόλη. Τα πολλά

αυτά προβλήματα μας είχαν καθυστερήσει πολλές φορές και αρκετές φορές βρεθήκαμε σε αδιέξοδο. Αυτό που καταφέραμε όμως είναι να έχουμε τις πληροφορίες που θέλαμε όσον αφορά τις ιδιοκτησίες.

Ένα από τα συμπεράσματα λοιπών είναι ότι, εάν το αρχείο είχε μια πιο δομημένη μορφή τότε η διαδικασία θα γινόταν πολύ πιο εύκολα και έτσι θα κερδίζαμε αρκετό χρόνο. Στο παράρτημα 2, παραθέτω κάποιες οδηγίες, έτσι ώστε σε περίπτωση που θα ήθελε κάποιος να ξαναπεράσει παρόμοια δεδομένα να έχουν μια καλύτερη δομή.

Προτείνω λοιπόν, για μελλοντική δουλειά, να συνεχιστεί η προσπάθεια για την ολοκλήρωση της εργασίας. Αυτό που πρέπει να γίνει για να ολοκληρωθεί η εργασία είναι, με βάση τις πληροφορίες που αποκτήσαμε, να υπολογιστούν τα εμβαδά της κάθε περιοχής και της κάθε ιδιοκτησίας έτσι ώστε να γνωρίζουμε και το μέγεθος της κάθε ιδιοκτησίας. Στη συνέχεια, με την βοήθεια κάποιου λογισμικού, να γραφτούν κανόνες για να γίνει η τρισδιάστατη μοντελοποίηση της πόλης. Παρόμοιες δουλειές για την κατασκευή κανόνων, είδαμε στις προηγούμενες εργασίες που έγιναν στο πανεπιστήμιο Κύπρου. Αυτοί οι κανόνες αφορούν περισσότερο την αρχιτεκτονική των σπιτιών την εποχή εκείνη.

Εάν ολοκληρωθεί η εργασία, τότε θα έχουμε ένα ολοκληρωμένο σύστημα το οποίο με κάποια παρόμοια δεδομένα εισόδου, θα μπορεί να μοντελοποιεί την οποιαδήποτε πόλη σε τρισδιάστατη μορφή.

Αυτό θα μπορούσε να χρησιμοποιηθεί για ιστορικούς σκοπούς κυρίως, γιατί εάν υπάρχουν αυτές οι πληροφορίες για μια παλιά πόλη, τότε θα μπορεί το σύστημα να την μοντελοποιεί, και θα μπορούμε να δούμε και τις άλλες πόλεις της Κύπρου στην εποχή εκείνη και να αποκτήσουμε περισσότερες πληροφορίες για τον πολιτισμό της Κύπρου χρόνια πριν.
Αυτό το σύστημα όμως, θα μπορούσε να χρησιμοποιηθεί, για οποιαδήποτε πόλη για την οποία έχουμε παρόμοιες πληροφορίες. Πιστεύω πως αυτή η δυνατότητα θα ενδιαφέρει αρκετές χώρες, γιατί με ένα τέτοιο σύστημα μπορούν να δουν σε τρισδιάστατη μορφή το πώς ήταν κτισμένες οι πόλεις στις οποίες ζουν, πολλά χρόνια πριν. Επίσης σε συνεργασία με αρχαιολόγους και αρχιτέκτονες, το σύστημα θα μπορεί να

βελτιστοποιηθεί για την καλύτερη δυνατή μοντελοποίηση των πόλεων και των αρχιτεκτονικών τους.

Εάν γίνουν αυτά, τότε στη συνέχεια θα μπορούσαν να μοντελοποιηθούν ολόκληρες χώρες και να μπορεί κάποιος να περιπλανιέται σε μια χώρα που υπήρχε πριν από πολλά χρόνια.

### 5.3.2 Εμπειρίες που Αποκτήθηκαν

Στα πλαίσια της διπλωματικής μου εργασίας, μου δόθηκε η ευκαιρία να αποκτήσω αμέτρητες εμπειρίες, οι οποίες είμαι βέβαιος πως στη συνέχεια της σταδιοδρομίας μου θα με βοηθήσουν σε πολλές περιπτώσεις. Για την ανάπτυξη της εργασίας ήταν αναγκαία να μελέτη αρκετής σχετικής βιβλιογραφίας κυρίως από ιστοσελίδες αλλά και για να ενημερωθώ για παρόμοιες εργασίες στο διαδίκτυο.

Για την ολοκλήρωση της εργασίας, χρειάστηκε να έρθω σε επαφή με πληθώρα λογισμικών όπως το AutoCAD, το 3D Studio max και άλλα πολλά, και έτσι να εμπλουτίσω τις γνώσεις μου γύρω από τέτοιου είδους λογισμικά. Χρειάστηκε να αναπτύξουμε δυο προγράμματα και έτσι μπορώ να πω ότι οι απόκτησα περισσότερες γνώσεις και προγραμματιστικές εμπειρίες. Ήρθα σε επαφή με λογισμικά που δεν μου δόθηκε ξανά η ευκαιρία και η διπλωματική μου εργασία ήταν μια μοναδική ευκαιρία.

Επίσης, έχω έρθει σε επαφή με πολλές μεθόδους και αλγόριθμους τους οποίους έπρεπε να μελετήσω και να τους υλοποιήσω. Επίσης μου δόθηκε η ευκαιρία να έρθω σε επαφή με την βιβλιοθήκη opnecv και να εμπλουτίσω τις γνώσεις μου σε αυτή, καθώς τη θεωρώ πολύ σημαντική στον τομέα της υπολογιστικής όρασης. Με τη βιβλιοθήκη επεξεργάστηκα την εικόνα του χάρτη, έχοντας από αυτή, αρκετή βοήθεια.

Η πιο σημαντική εμπειρία που απεκόμισα ήταν η αντιμετώπιση όλων των προβλημάτων που συνάντησα κατά τη διάρκεια της υλοποίησης της εργασίας. Σε πολλές περιπτώσεις τα προβλήματα αυτά ήταν πολύ σοβαρά και μας έκαναν να αλλάξουμε προσέγγιση της εργασίας και πιστεύω πως η επίλυση των προβλημάτων αυτών, με τη σωστή

καθοδήγηση του επιβλέποντος καθηγητή μου Δρ. Γιώργο Χρυσάνθου, ήταν η πιο σημαντική.

Τέλος πιστεύω αυτή η εργασία, παρά τα προβλήματα που αντιμετώπισα, μου έδωσε πολλά μαθήματα και γνώσεις οι οποίες θα με βοηθήσουν στα μέγιστα στις μεταπτυχιακές μου σπουδές και γενικότερα στην μελλοντική επαγγελματική μου σταδιοδρομία.

# Βιβλιογραφία


[1] C. Xu and Jerry L. Prince, Snakes, Shapes, and Gradient Vector Flow, EEE TRANSACTIONS ON IMAGE PROCESSING. volume 7. number 3 pages 359-369. 1998.

[2] P.Muller, P.Wonka, S.Haegler, A.Ulmer and Luc Van Gool, Procedural Modeling of Buildings, ACM Transactions on Graphics. volume 25. number 3 pages 614-623. 2006. Proceedings of SIGGRAPH 2006

[3] Α. Ευθυμίου, Δημιουργία Τρισδιάστατων Εικονικών Μοντέλων Ξεκινώντας από Χάρτες και Φωτογραφίες. Μοντελοποίηση των Εντός των Τειχών Λευκωσίας , Ατομική Διπλωματική Εργασία, 2003.

[4] Ε. Ηλιάδη, Ο μαχαλάς της Λευκωσίας, Taht el Kale – μέσα από την πρώτη πλήρη κτηματολογική εγγραφή μετά το τέλος της Οθωμανικής Περιόδου στη Κύπρο, Διδακτορική Διατριβή .Ιούνιος 2008.

[5] Σ. Μιχαήλ, Διαδικαστική μοντελοποίηση της παλιάς Λευκωσίας του 19$^{ου}$ αιώνα, Ατομική Διπλωματική Εργασία, 2009.

[6] Procedural Website, http://www.procedural.com:9099/help/index.jsp?topic=/com.procedural.cityengine.help/html/toc.html

[7] The C++ Resources Network, http://www.cplusplus.com/

[8] Computer Vision Lab, http://www.vision.ee.ethz.ch/

# Παράρτημα Α

Σε αυτό το παράρτημα παραθέτω τον κώδικα που γράφτηκε στην γλώσσα προγραμματισμού java που παίρνει σαν είσοδο το αρχείο και υπολογίζει τις πληροφορίες για τις ιδιοκτησίες.

```
import java.awt.BorderLayout;
import javax.rmi.CORBA.Util;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import java.awt.Color;
import java.awt.Dimension;
import java.io.BufferedReader;
import java.io.Console;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.io.Writer;

import javax.swing.*;
import java.util.Random;
import javax.swing.JFrame;

import org.apache.commons.lang.StringUtils;
import org.omg.Messaging.SYNC_WITH_TRANSPORT;

import java.io.File;
```

```
import java.lang.*;
import java.nio.Buffer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import java.util.Map.Entry;
import java.util.logging.ConsoleHandler;

import java.util.*;

//i clasi Omades
public class omades {
/**********************************************METAVLITES*************
************************************************************************/
        static private String[][] file= new String [300][5];  //o disdiastatos pinakas opou tha apothikeftei to arxeio

        static private int coun=0;                          //enas metritis opou antiprsosoprvei ton arithmo

                                //twn grammwn tou arxeiou

        static private HashMap<String, HashSet<String>> omades = new HashMap<String, HashSet<String>>(); //ena hash table opou krata


                                //tis omades apo tous geitones
```

```
        static private Map<String[], Boolean> SimilarStrings = new HashMap <String[],
Boolean>();    //ena hash table opou kratoume ola ta zevgaria apo tous idioktites


        //opou exoun 3anaerwtithei ean einai ta idia

/*********************************************METHODOI*************
************************************************************/

        //afti i methodos sinenwnei dio omades se mia
        static private void combine(String table[], String holder){
     String key=null;
     int k=0;
    for(int i=0;i<table.length;i++){
          for (String value : omades.get(table[i])){
          if(k==0){
                  key=createNewTeam(value);
           }
          else {
                  omades.get(key).add(value);
          }
         k=1;
        }
                  omades.remove(table[i]);
    }
    omades.get(key).add(holder);
   }




/*****************************************************************
*************************************************************/
```

```
//i sinartisi afti ipologizei gia kateh geitona mias odioktisias ton monadiko arothmo pou antistixei se afto
  static private String neighReplace(String n, int holder) throws IOException{

        int theseis[]= new int[50];
        int counter=0;
        boolean flag=false;
        if(n.contains("succession of")){
                n=n.replace("succession of", "");
                n=n.trim();
                for(int i=0;i<coun;i++,flag=false){
                        String line[] = file[i];
                                line[3]=line[3].trim();
                                n=n.trim();
                                if(line[3].contains(",")){
                                        String multi_holders[]=line[3].split(",");
                                        for(int j=0; j<multi_holders.length;j++){
                                                if(multi_holders[j].contains(n)){
                                                        if(flag==false){
                                                        theseis[counter]=i;
                                                        counter++;
                                                        flag=true;
                                                        }
                                                }
                                        }
                                }
                }
        if(counter==0){
                return n;
        }
        else if(counter==1){
                //System.out.println("Succession found and it is"+ file[theseis[0]][3]);
```

```
                return file[theseis[0]][1] + " " + file[theseis[0]][2];
        }
        else if(counter>1){
                for(int i=0;i<counter;i++){
                 String neighs[]= file[theseis[i]][4].split(",");
                 for(int j=0;j<neighs.length;j++){
                        if(neighs[j].contains(file[holder][3])){
                       //System.out.println("Succession found and it is"+
file[theseis[i]][3]);
                                return file[theseis[i]][1] + " " + file[theseis[i]][2];
                        }
                                else if(similar_check(neighs[j],file[holder][3])==true){
                                return file[theseis[i]][1] + " " + file[theseis[i]][2];
                        }
                }
         }
        }

   }
        else if(n.contains("and others")){
        //System.out.println("And others found");
        n=n.replace("and others", "");
                n=n.trim();
                for(int i=0;i<coun;i++,flag=false){
                        String line[] = file[i];
                        line[3]=line[3].trim();
                        n=n.trim();
                        if(line[3].contains(",")){
                                String multi_holders[]=line[3].split(",");
                                for(int j=0; j<multi_holders.length;j++){
                                        if(multi_holders[j].equals(n)){
                                                if(flag==false){
                                                theseis[counter]=i;
```

```
                                        counter++;
                                        flag=true;
                                        }
                                }
                                else if(similar_check(multi_holders[j], n)){
                                        if(flag==false){
                                                theseis[counter]=i;
                                                counter++;
                                                flag=true;
                                                }
                                }
                        }
                }

}
        if(counter==0){
        return n;
}
else if(counter==1){
        //System.out.println("Others found and it is"+ file[theseis[0]][3]);
        return file[theseis[0]][1] + " " + file[theseis[0]][2];
}
else if(counter>1){
        for(int i=0;i<counter;i++){
         String neighs[]= file[theseis[i]][4].split(",");
         for(int j=0;j<neighs.length;j++){
                 if(neighs[j].equals(file[holder][3])){
                //System.out.println("Others found and it is"+ file[theseis[i]][3]);
                        return file[theseis[i]][1] + " " + file[theseis[i]][2];
                 }
                         else if(similar_check(neighs[j],file[holder][3])==true){
                        return file[theseis[i]][1] + " " + file[theseis[i]][2];
                 }
```

```
                }
            }
            }

            }
            else{
                if(n.contains("wife")){
                    //System.out.println("hahahahahahahahah "+ n);
                        int j = n.indexOf("wife", 0);
                        n = n.substring(0, j);
                        n = n.trim();
                        //System.out.println("huhuhuhuhuhuhuhuh "+ n);
                   }
//                  if(n.contains("of")  &&  !(n.contains("Church")  ||
n.contains("Government") || n.contains("Municipality"))){
//                      int j = n.indexOf("of", 0);
//                          n = n.substring(0, j);
//                          n = n.trim();
//                  }
            for(int i=0;i<coun;i++){
                        try {
                            String line[] = file[i];
                            line[3]=line[3].trim();
                            n=n.trim();
                        if(n.equals(line[3])){
                            theseis[counter]=i;
                            counter++;
                        }
                        else if(similar_check(n,line[3])==true){
                            theseis[counter]=i;
                            counter++;
                        }
                    }
```

```
                        catch(Exception EX){
                                continue;
                        }
    }
        if(counter==0){
          // System.out.println("No match with "+n);
           return n;
        }
        else if(counter==1){
          return file[theseis[0]][1] + " " + file[theseis[0]][2];
        }
        else if(counter>1){
           for(int i=0;i<counter;i++){
                String neighs[]= file[theseis[i]][4].split(",");
                for(int j=0;j<neighs.length;j++){
                        if(neighs[j].equals(file[holder][3]))
                                return file[theseis[i]][1] + " " + file[theseis[i]][2];
                        else if(similar_check(neighs[j],file[holder][3])==true){
                                return file[theseis[i]][1] + " " + file[theseis[i]][2];
                        }
                }
           }
        }
          }
           return n;
    }

/**********************************************************************
**********************************************************/

    //i methodos afti kanei kapoies allages sto arxeio opou tha voithisoun stin sinexeia
        static private void replace(){
```

```
                for(int i=0;i<coun;i++){
                        try {
                        String line[]=file[i];
                        if(line[3].contains("wife")){
                                int j = line[3].indexOf("wife", 0);
                                line[3] = line[3].substring(0, j);
                                line[3] = line[3].trim();
                    }
//                      if(line[4].contains("wife")){
//                              System.out.println("wife found" + line[4]);
//                              int j = line[4].indexOf("wife", 0);
//                              line[4] = line[4].substring(0, j);
//                              line[4] = line[4].trim();
//                  }

//                      if(line[3].contains("of")  &&  !(line[3].contains("Church")  ||
line[3].contains("Government") || line[3].contains("Municipality"))){
//                              int j = line[3].indexOf("of", 0);
//                              line[3] = line[3].substring(0, j);
//                              line[3] = line[3].trim();
//                  }
                        line[4] = line[4].replace("&", ",");
                        line[4] = line[4].replace("\"", "");
                        line[4] = line[4].replace(";", ",");
                        file[i]=line;
                }
                        catch(Exception EX){
                                continue;
                        }
                }

        }
```

```
/*****************************************************************************
*************************************************************/

  //afti i maethodos elenxei ean gia ena zevgari idioktitwn exei 3anarwtithei ean einai ta
idia
  static private int asked_before(String s[]){

        for(Object str[] : SimilarStrings.keySet()){
              if((s[0].equals(str[0]) && s[1].equals(str[1])) || (s[0].equals(str[1]) &&
s[1].equals(str[0])) ){
                    if(SimilarStrings.get(str).booleanValue())
                          return 1;
                    else
                          return 0;
              }
        }
        return 2;
}


/*****************************************************************************
*************************************************************/

  //i methodos afti kanei tin diadikasia opou o xristis tha apofasisei ean dio idioktitws
parolo pou einai diaforetika grammenes sto arxeio einai oi idioi
              static private boolean similar_check(String A, String B) throws
IOException{
                    A.trim();
                    B.trim();
                    String str[] = new String[2];
                    str[0]=A;
                    str[1]=B;
                    int flag=asked_before(str);
```

```
                    if(flag==2){
                    int diff=StringUtils.getLevenshteinDistance(A, B);

                    if(diff<=3)       // ean i diafora twn dio strings einai mikroteri apo
3 tote tha ta theorisoume ta idia
                              return true;

                    else if(diff<=4){      //ean i diafora einai 4  tote tha erwtithei o
xristis ean thelei na theorithoun ta idia
                                        System.out.println("Do you want to consider these
two strings as same?");
                                         System.out.println(A + "\n" +B);
                                         System.out.println("Yes(y)/No(n)");

                                         BufferedReader br = new BufferedReader(new
InputStreamReader(System.in),1);
                                         String readLine = br.readLine();
                                         System.out.println(readLine);
                                        if(readLine.endsWith("y")                    ||
readLine.endsWith("Y")) //ean o xristis dwsei tin epilogi nai
                                                  {
                                                  SimilarStrings.put(str, true);
                                                  return true;
                                                   }
                                        else          if(readLine.endsWith("n")          ||
readLine.endsWith("N"))
                                        {
                                                  SimilarStrings.put(str, false);
                                                  return false;
                                        }
               }
                    SimilarStrings.put(str, false);
                 return false;
```

```
                }
                else{
                        if(flag==1)
                                return true;
                        else
                                return false;
                }
        }

/**********************************************************************
**********************************************************/

        //i sinartisi afti dimiourgei mia kainouria omada apo geitones
        static private String createNewTeam(String holder) {
                HashSet<String> newSet = new HashSet<String>();
                newSet.add(holder);
                omades.put(Integer.toString((omades.size() + 1)), newSet);
                return Integer.toString(omades.size());
                //System.out.println(newSet);
        }

/**********************************************************************
**********************************************************/

        //i sinartisi afti elexei ean aftos o holder prepei na topothetithei se kapoia
omada.ean nai tote tin topothetei
        static private void addInHashTable(String[] gitones, String holder) throws
IOException {

                int count = 0, flag = 0;
                String[] k = new String[gitones.length];
                 for(int i = 0;i < k.length;i++){
                        k[i]="a";
```

```
 }
if(omades.isEmpty()){
        createNewTeam(holder);
}
else{
for(int i = 0;i < gitones.length; i++,flag = 0){
        for (Object key : omades.keySet()) {
                for (String value : omades.get(key)){
                        if (value.equals(gitones[i])){
                                String p=(String) key;
           for(int j=0;j<k.length;j++){
                if(k[j].equals(p)){
             flag=1;
           }
          }
        if(flag==0){
                k[count] = p;
                        count++;
          }
           flag=0;
                        }

          }
        }
}
if(count==0){
        createNewTeam(holder);
}

else if(count==1){
        omades.get(k[0]).add(holder);
}
else if(count>1){
```

```
                omades.get(k[0]).add(holder);
                combine(k,holder);
            }
            }
        }

/**********************************************************************
**********************************************************/

 //Stin methodo afti diavazoume apo to arxeio kai to fortonoume se ena disdiastato pinaka apo strings
        static void read(String textfile) {
            String CurrLine;

            try {
                FileInputStream fstream = new FileInputStream(textfile);
                DataInputStream in = new DataInputStream(fstream);
                BufferedReader br = new BufferedReader(new InputStreamReader(in));
                CurrLine = br.readLine();

                while (CurrLine != null) {

                        file[coun]=CurrLine.split("\t");

                        coun++;
                        CurrLine = br.readLine();


                }
                in.close();
            }
```

```
                catch (IOException e) {
                        System.err.println("Error:" + e.getMessage());
                        System.exit(-1);
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

/**
 *                                      @throws                                 IOException
**********************************************************************
**********************************************************/

 static void FindItsNeigh(String name, String number , String k) throws IOException{

          for (Object key : omades.keySet()) {
                        for (String value : omades.get(key)){
                    if(FoundTeam(value,name)){
         //ean i omada tou vrethike
                         // System.out.println("The key is "+key);
                          omades.get(key).add(number);
                          return;
                    }
                        }
                }

 }


 /**p
 *                                      @throws                                 IOException
**********************************************************************
**********************************************************/
```

```
static boolean FoundTeam (String value , String n) throws IOException{

        for(int i=0; i < coun; i++){
         String line[] = file[i];
         String holder = line[1] + " " + line[2];
         if(holder.equals(value)){
                if(line[4].contains(",")){
                String Neighs[] = line[4].split(",");
                for(int j = 0; j < Neighs.length; j++){
                        if(Neighs[j].equals(n)){
                                return true;
                        }
                        else if(similar_check(Neighs[j],n)==true){
                                return true;
                        }
                }
                }
                else{
                        //ean o singekrimenos holder exei mono ena geitona
                        if(line[4].equals(n)){
                                return true;
                        }
                        else if(similar_check(line[4],n)==true){
                                return true;
                        }
                }
         }
        }

        return false;
}
```

```
/**********************************************************************
**********************************************************/
        //i main methodos
        public static void main(String[] args) throws IOException {

                String line[]=null;

                read("info.txt"); //edw kaloume afti ti sinartisi gia na diavasome to arxeio

                replace();   //I sinartisi afti kaleitai edw gia na ginoun
                                        //kapoies allages sto arxeio(opws ta simvola '&'
na ginoun ',' k.a)
//              for(int i=0;i<file.length;i++){
//                      for(int j = 0 ; j < 5 ; j++){
//                      System.out.print(file[i][j]+ "  ");
//
//              }
//                      System.out.println();
//              }
                for(int i=0;i<=coun;i++){
        try {
        line=file[i];
        String holder = line[1] + " " + line[2];
        String neighTable[] = line[4].split(",");
                for (int k = 0; k < neighTable.length; k++) {
                        neighTable[k] = neighTable[k].trim();
                        neighTable[k] = neighReplace(neighTable[k],i);
                }
                //    for(int j=0;j<neighTable.length;j++)
                        //System.out.print(neighTable[j]+ ", ");
                  addInHashTable(neighTable,holder);
```

```
            //  System.out.println();
    }
    catch (Exception ex) {
                                continue;
                        }
}



  String name=null;
  String number=null;
  Object rem[] = new Object[omades.size()];
  int rem_counter=0;
 int k=0,count=0;
  for (Object key : omades.keySet()) {
     for (String value : omades.get(key)){
              number=value;
               k++;
     }
     if(k==1){
               count++;
               for(int i=0; i<coun ; i++){
                        String holder = file[i][1]+ " " + file[i][2];
                        if(number.equals(holder)){
                                 name = file[i][3];
                        break;
                        }
               }
            FindItsNeigh(name,number,key.toString());
            rem[rem_counter]=key;
            rem_counter++;
     }
```

```
            k=0;
        }

        for(int i=0; i< rem.length; i++){     // edw afairoume tin omada opou vriskontan oi
idioktites pou den itan se kapoia omada
            omades.remove(rem[i]);
        }

        System.out.println(omades);
         }

}
```

# Παράρτημα Β

Σε αυτό το παράρτημα παραθέτω τις οδηγίες για την σωστή και δομημένη εγγραφή στο αρχείο εισόδου έτσι να γίνεται πιο αποδοτικά η επεξεργασία των δεδομένων.

**Πρώτη στήλη:**
Εδώ θα αναγράφεται ο αύξον αριθμός της ιδιοκτησίας.

**Δεύτερη στήλη:**
Εδώ θα αναγράφεται ο αριθμός σχεδίου.

**Τρίτη στήλη:**
Εδώ θα αναγράφεται ο αριθμός οικοπέδου.

**Τέταρτη στήλη:**
Εδώ θα αναγράφεται το όνομα του ιδιοκτήτη μόνο(χωρίς κάποιες επιπρόσθετες πληροφορίες)

**Πέμπτη στήλη:**
Εδώ θα αναγράφονται οι όποιες επιπρόσθετες πληροφορίες υπάρχουν για τον ιδιοκτήτη (π.χ wife of costi parperi).

**Έκτη Στήλη**
Εδώ θα αναγράφονται οι γείτονες της ιδιοκτησίας χωρισμένοι με τον χαρακτήρα «,». Σε περίπτωση που υπάρχουν περισσότερες πληροφορίες για ένα γείτονα, τότε μετά το όνομά του να τοποθετείται ο χαρακτήρας «-» και ακολούθως οι πληροφορίες.

**Έβδομη στήλη:**
Εδώ θα αναγράφεται τι κτήριο είναι η ιδιοκτησία.(π.χ κατάστημα, σπίτι κλπ).

**Όγδοη Στήλη:**
Εδώ θα αναγράφεται ο αριθμός των δωματίων της ιδιοκτησίας.

**Ένατη Στήλη:**

Εδώ θα αναγράφεται ο αριθμός των Μπαλκονιών της ιδιοκτησίας(εάν υπάρχουν). Σε περίπτωση που δεν υπάρχει αυτή η πληροφορία να αναγράφεται ο χαρακτήρας «-»

**Δέκατη στήλη:**

Εδώ θα αναγράφονται άλλες πληροφορίες για την ιδιοκτησία όπως εάν έχει αυλή, κουζίνα, τουαλέτα, δέντρα κλπ.

**Ενδέκατη στήλη:**

Εδώ θα αναγράφεται το μέγεθος της ιδιοκτήίας.