

Γεώργιος Τοκατλίδης

Α.Μ : 58352

**Όραση Υπολογιστών – Εργασία 2**

**ΜΕΡΟΣ Α’**

**Γενική επισκόπηση**

Το script αυτό έχει σχεδιαστεί για να εκτελεί συγχώνευση εικόνων χρησιμοποιώντας την ανίχνευση χαρακτηριστικών και τον μετασχηματισμό ομογραφίας. Ας αναλύσουμε τα κύρια στοιχεία του κώδικα:

**Συναρτήσεις Συγχώνευσης Εικόνων:**

**image\_stitching**: Παίρνει τα αντιστοιχισμένα σημεία ενδιαφέροντος και τους περιγραφείς, βρίσκει τον μετασχηματισμό ομογραφίας συγκεντρώνει δύο εικόνες. Χειρίζεται τα όρια, απορρίπτοντας τα περιττά μέρη, δηλαδή τα μαύρα περιθώρια.

**cross\_checking**: Συγκρίνει δύο σύνολα αντιστοιχίσεων για εύρεση διασταυρούμενων αντιστοιχίσεων και προαιρετικά τις εμφανίζει χρησιμοποιώντας τη βιβλιοθήκη OpenCV.

**feature\_matching**: Αντιστοιχεί σημεία ενδιαφέροντος και περιγραφείς μεταξύ δύο εικόνων χρησιμοποιώντας έναν προσαρμοσμένο έλεγχο αναλογίας απόστασης. Δίνεται επιλογή για εμφάνιση ή μη των αντιστοιχίσεων.

**Συνάρτηση Ανάγνωσης Εικόνας:**

**img\_reader:** Διαβάζει μια εικόνα, ανιχνεύει τα σημεία ενδιαφέροντος και υπολογίζει τις περιγραφές χρησιμοποιώντας είτε τον αλγόριθμο SIFT ή SURF.

Προαιρετικά εμφανίζει την αρχική εικόνα.

**Κύριο Πρόγραμμα:**

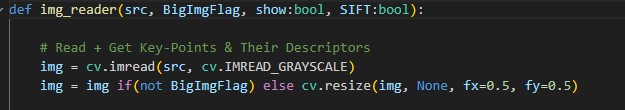
Διαβάζει τέσσερις εικόνες και εκτελεί τη συγχώνευση εικόνων σε πολλά βήματα:

* Αντιστοιχεί και συγχωνεύει τις εικόνες 1-2.
* Αντιστοιχεί και συγχωνεύει τις εικόνες 3-4.
* Αντιστοιχεί και συγχωνεύει τα αποτελέσματα των προηγούμενων βημάτων (εικόνα 1-2 και εικόνα 3-4).

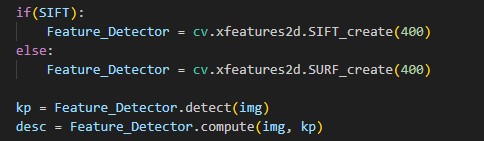
Επιλογή Ανιχνευτή Χαρακτηριστικών: Το σενάριο επιτρέπει στον χρήστη να επιλέξει μεταξύ των ανιχνευτών χαρακτηριστικών SIFT και SURF με το SIFT flag.

**Ανάλυση κώδικα**

**img\_reader:** Διαβάζειτην εικόνα σε διαβάθμιση του γκρι. Εάν η σημαία BigImgFlag είναι True, η εικόνα μειώνεται στο μισό με τη χρήση cv.resize. Αυτό χρησιμοποιείται για εικόνες μεγάλων διαστάσεων που δε μπορούν να χωρέσουν σε παράθυρο προβολής. Στη περίπτωσή μας χρησιμοποιείται για τις εικόνες που εισάχθηκαν από εμένα.



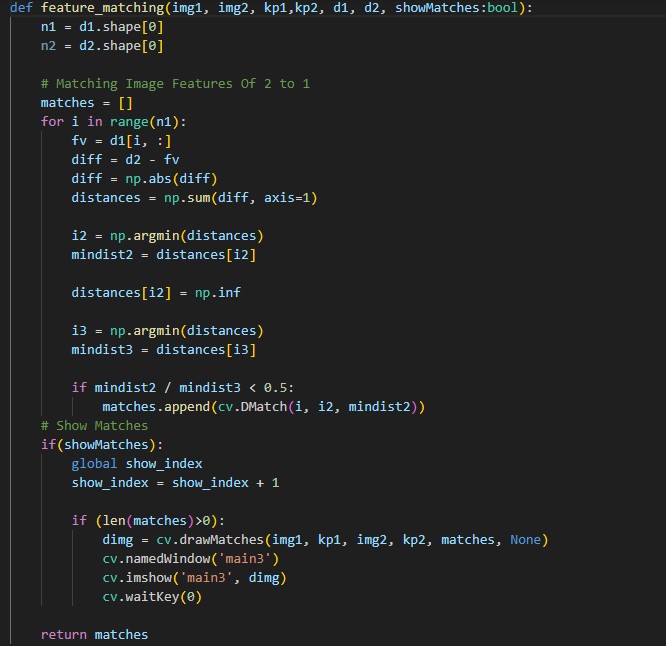
Εάν η σημαία SIFT είναι True, δημιουργεί έναν ανιχνευτή SIFT χαρακτηριστικών με ένα κατώφλι του 400, αλλιώς δημιουργεί έναν ανιχνευτή SURF χαρακτηριστικών με το ίδιο κατώφλι.



Εάν η σημαία show είναι True, δημιουργεί ένα παράθυρο με το όνομα 'main' + αριθμός εμφάνισης (show\_index) και εμφανίζει την αρχική εικόνα. Αυτό χρησιμοποιείται για επίδειξη των αποτελεσμάτων.

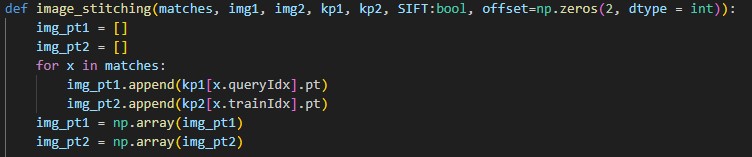
**feature\_matching**: Υπολογίζει τον αριθμό των κλειδιών στις δύο εικόνες, n1 για την πρώτη εικόνα και n2 για τη δεύτερη. Έπειτα με μια δομή επανάληψης εξετάζεται για κάθε σημείο-κλειδί της 1ης εικόνας η διαφορά του περιγραφέα του με τους περιγραφείς της 2ης εικόνας (σε απόλυτη τιμή). Υπολογίζει το άθροισμα των απόλυτων διαφορών για κάθε κλειδί της δεύτερης εικόνας, δημιουργώντας ένα διάνυσμα αποστάσεων (distances). Βρίσκει τον δείκτη της ελάχιστης απόστασης. Αποθηκεύει την ελάχιστη απόσταση στη μεταβλητή. Θέτει την απόσταση του εντοπισμένου κλειδιού στο άπειρο ώστε να μην επιλεγεί ξανά (να εξασφαλιστεί ότι δε θα θεωρηθεί επανειλημμένα η μικρότερη απόσταση). Βρίσκει τον δείκτη δεύτερης ελάχιστης απόστασης. Αποθηκεύει τη δεύτερη ελάχιστη απόσταση.

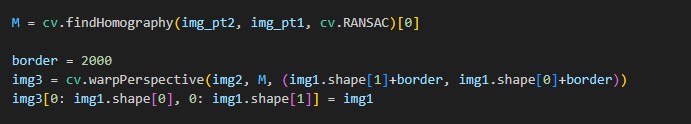
Αν ο λόγος πρώτης προς δεύτερης ελάχιστης απόστασης είναι μικρότερος από 0.5, τότε αποκτούμε ένα αντικείμενο cv.DMatch στη λίστα matches. Επίσης δίνεται πάλι η δυνατότητα εμφάνισης των δύο εικόνων με μαρκαρισμένα τα συζευγμένα χαρακτηριστικά τους.

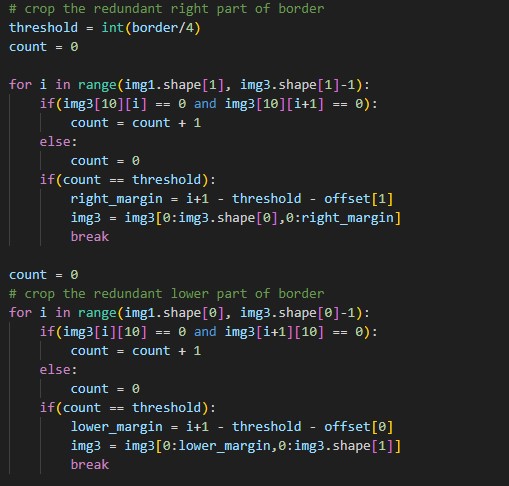


**Cross\_checking:** Σε μια νέα λίστα προστίθενται τα σημεία-κλειδιά της πρώτης εικόνας τα οποία έχουν συζευχθεί με τα αντίστοιχα σημεία-κλειδιά της 2ης εικόνας και τα ίδια σημεία κλειδιά της 2ης εικόνας έχουν αντιστοιχηθεί στα σημεία κλειδιά της 1ης εικόνας.

**image\_stitching**: Σε δύο λίστες, μια για κάθε εικόνα, προσθέτουμε τις συντεταγμένες κάθε σημείου ενδιαφέροντος (.pt). Το x.\_\_\_\_Idx δίνει τον δείκτη στο set kp για το οποίο παίρνουμε το x match.



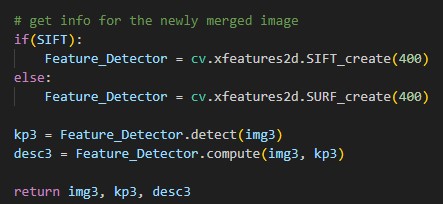
Με τη συνάρτηση findHomography αλγοριθμικά βρίσκεται (με το RANSAC) ο πίνακας μετασχηματισμού που πρέπει να εφαρμοστεί στη 2η εικόνα για να ταιριάξει με τη 1η. Έπειτα τις συνενώνει με ένα παράθυρο όσο το αρχικό συν ένα περιθώριο 2000 pixel και στους 2 άξονες. Στο τέλος εφαρμόζουμε μια μάσκα, δηλαδή σε ένα τμήμα της φωτογραφίας, το οποίο ταυτίζεται σε διαστάσεις με τη 1η εικόνα, απαιτούμε να υπάρχει το τμήμα της 1ης εικόνας και όχι της 2ης που συνταιριάχθηκε.

Tο υπόλοιπο τμήμα του κώδικα αυτής της μεθόδου χειρίζεται τα περιθώρια που προκύπτουν από το αρχικό περιθώριο που δίνουμε, το οποίο εσκεμμένα τίθεται μεγάλο, για να εξασφαλίσουμε ότι οι διαστάσεις της νέας εικόνας μετά την συνένωση θα είναι αρκετές για να απεικονίσουν όλη τη πληροφορία. Αυτό γίνεται ως εξής. Κάνουμε τη παραδοχή ότι στην εικόνα δεν πρόκειται να έχουμε το απόλυτο μαύρο για πολλά διαδοχικά pixel. Τον όρο ‘πολλά’ τον ποσοτικοποιούμε με τη μεταβλητή threshold. Μέσα στις δομές επανάληψης (μια για τον άξονα x και μια για τον y) ελέγχουμε αν το ν-οστό στοιχείο και ν+1 στοιχείο είναι μαύρο. Σε αυτή τη περίπτωση αυξάνουμε έναν μετρητή. Αν ο μετρητής φτάσει το threshold τότε θεωρούμε πως βρισκόμαστε εκτός της εικόνας και διατρέχουμε το περισσευούμενο μαύρο περιθώριο. Για να γλυτώσουμε επαναλήψεις λαμβάνουμε για αριστερό όριο στον άξονα x και άνω όριο στον άξονα y, τις διαστάσεις της εικόνας εκμεταλλευόμενοι της τελευταίας εντολής που φαίνεται στο παραπάνω snapshot.



Το offset χρησιμοποιείται για να κόψει εναπομείνουσες μαύρες λωρίδες οι οποίες προκύπτουν επειδή η 2η εικόνα σε περίπτωση κάμψης της που φαίνεται σαν περιστροφή κατά τον μετασχηματισμό της, δημιουργεί ανισοτροπικότητα στο μαύρο περιθώριο που μπορεί να εμφανιστεί στην νέα εικόνα. Για αυτό εισάγεται το offset το οποίο κόβει περαιτέρω ένα μικρό κομμάτι του περιθωρίου το οποίο φαινομενικά θυσιάζει και τα όρια της εικόνας.

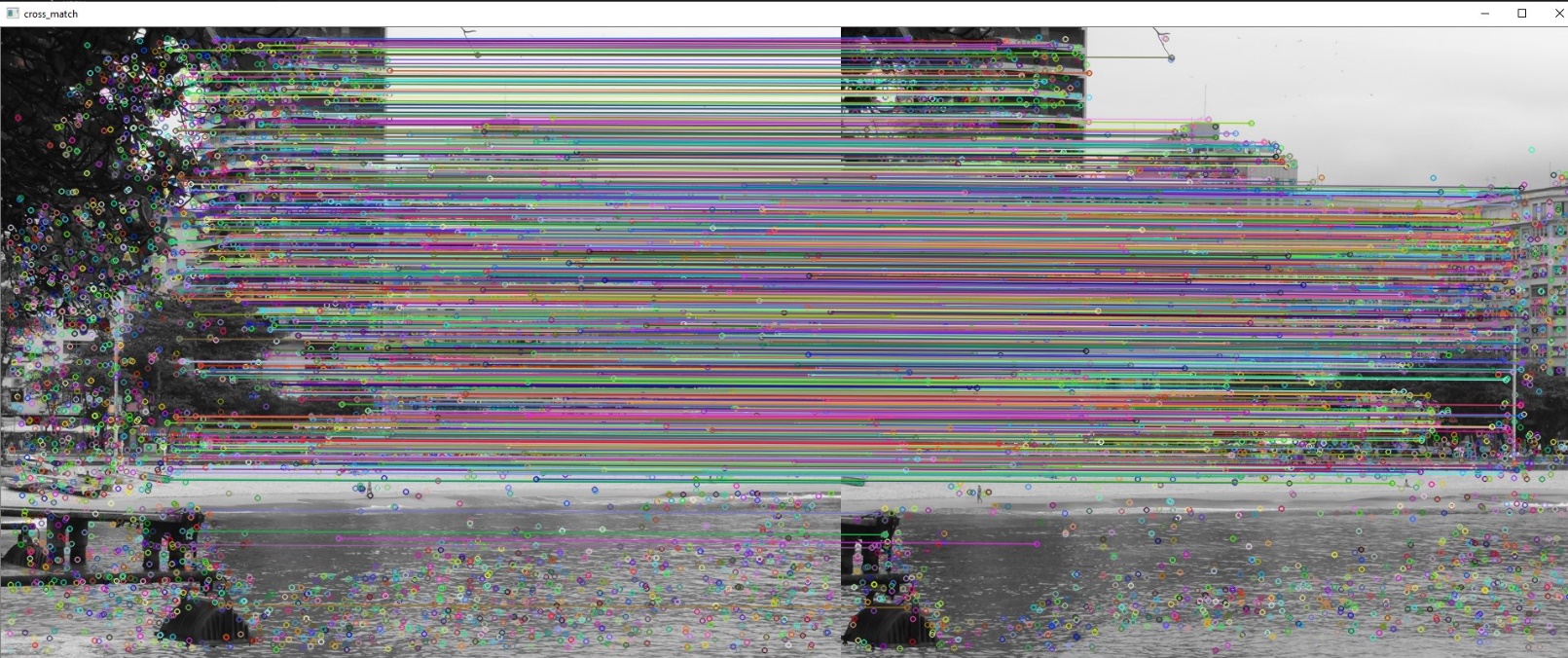
Όπως γίνεται αντιληπτό όσο και να μικρύνουμε το threshold θα έχουμε το ίδιο ακριβώς φαινόμενο, διότι με τον τρόπο που έχει σχηματιστεί ο αλγόριθμος, εμφανίζεται εξάρτηση μόνο από πού ξεκινάμε τη δομή επανάληψης, δηλαδή αν πάρουμε για παράδειγμα την απαλοιφή του δεξιού περιθωρίου στην εικόνα, εξαρτάται ποια γραμμή (ποιο y) έχουμε λάβει για να πραγματοποιήσουμε τη μέτρηση. Επειδή την επιλογή τη κάνουμε τυχαία εισάγουμε αυτό τον όρο για να εξασφαλίσουμε πως δεν θα έχουμε ασυνέχειες χωρίς να χάνουμε σημαντικό κομμάτι της πληροφορίας. Αυτό το βήμα είναι απαραίτητο βέβαια, μόνο για τη περίπτωση που θέλουμε να συνενώσουμε τη νέα εικόνα με 3η εικόνα. Σε αυτή τη περίπτωση συνεπώς δε χάνεται η πληροφορία από τα όρια τελικά, διότι το κομμένο όριο της εικόνας που χάθηκε λόγω του offset λαμβάνεται πίσω από τη 3η εικόνα.

Τέλος λαμβάνουμε, τα σημεία ενδιαφέροντος και τους περιγραφείς της νέας εικόνας

**Αποτελέσματα**

**Εικόνες RIO:** Ο αριθμός των Matches εκατέρωθεν για κάθε συνδυασμό, βγαίνει ταυτόσημος ή πολύ κοντά. Κύριος παράγοντας αυτού είναι το ότι συγκρίνουμε τις 2 μικρότερες αποστάσεις για κάθε match που εμφανίζει η εικόνα. Σύμφωνα με το paper για τον αλγόριθμο SIFT, αυτή η τεχνική απαλείφει 90% των false matches και το 5% των πραγματικών, άρα εκεί αποδίδουμε και την εγγύτητα των τιμών. Ως αποτέλεσμα έχουμε και ο αριθμός των cross\_matches να είναι κοντά στον αριθμό Matches ενός συνδυασμού.

Η κύρια διαφορά που φάνηκε μεταξύ των 2 αλγορίθμων είναι το πλήθος των αντιστοιχίσεων που έγιναν. Στον αλγόριθμο SURF έχουμε αισθητά μεγαλύτερο πλήθος αντιστοιχίσεων, οπότε και επαληθευμένων αντιστοιχίσεων. Ως συνέπεια έχουμε καλύτερη σύζευξη (μεταξύ 1ης και 2ης εικόνας), αλλά πολύ πιο αργό χρόνο εκτέλεσης. Παρακάτω ακολουθούν τα αποτελέσματα cross matches για τις εικόνες 1 και 2 και το τελικό αποτέλεσμα συνένωσης όλων των εικόνων για αναπαράσταση των παραπάνω.

SURF:

SIFT:





**Εικόνες Γηπέδου:** Τις ίδιες ακριβώς διαφορές παρατηρούμε για τους 2 αλγορίθμους και εδώ. Εδώ ωστόσο ο αλγόριθμος SURF δεν εντοπίζει υπερβολικά πολλά περιττά σημεία με αποτέλεσμα να είναι το ίδιο γρήγορος με τον SIFT. Η συνένωση εικονών εμφανίζεται προβληματική σε ορισμένα σημεία αλλά ίσως να εξηγείται διαισθητικά.

Αρχικά πρέπει να σημειωθεί ότι οι εικόνες είναι snapshots από βίντεο. Επιλέχθηκαν αυτές για τον λόγο ότι αποτελούν πρόκληση ακόμα κ σε ανεπτυγμένες εφαρμογές η δημιουργία πανοραμάτων σε ημι-κλειστούς χώρους με καμυλώσεις. Βέβαια αυτή η τοποθέτηση προέρχεται καθαρά από προσωπική εμπειρία με τη χρήση της λειτουργίας πανοράματος σε κάμερα κινητών τηλεφώνων.

Ένα χαρακτηριστικό τέτοιων εφαρμογών είναι πως απαιτείται η κίνηση της κάμερας να είναι αργή και να μην εμπεριέχει περιστροφές γύρω από άξονα άλλο εκτός του ατόμου. Οι λόγοι είναι πολλοί αλλά αυτός που αφορά εμάς είναι ο αριθμός των καρέ. Μια πιστότερη απεικόνιση πανοράματος γίνεται με όσο δυνατόν περισσότερα καρέ μπορούμε να έχουμε από μια σκηνή, διότι όσο περισσότερα είναι τα καρέ τόσο περισσότερο τείνουν να ταυτιστούν 2 διαδοχικές εικόνες μεταξύ τους, άρα και να εμφανίσουν περισσότερα πραγματικά κοινά σημεία.

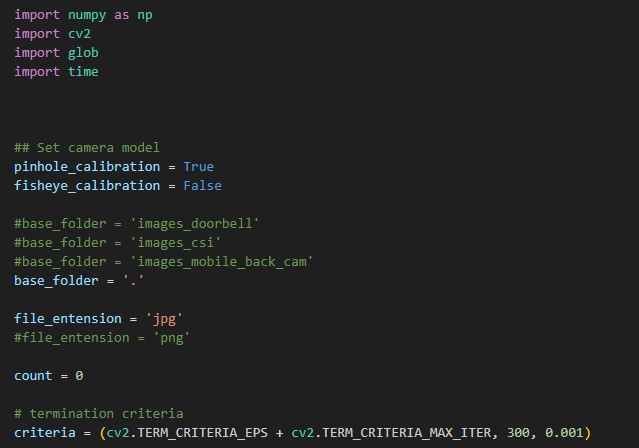
**ΜΕΡΟΣ Β’**

**Γενική επισκόπηση**

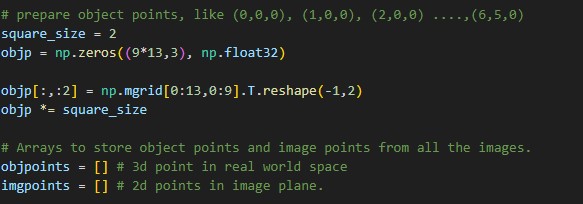
Το μέρος αυτό αφορά απαρτίζεται από 3 τμήματα. Την βαθμονόμηση κάμερας με χρήση κάποιου προτύπου, την αφαίρεση παραμόρφωσης με χρήση τον πίνακα βαθμονόμησης και παραμόρφωσης που εισάγει η κάμερα και τέλος την εκτίμηση βάθους σε μια σκηνή με οδηγό 2 φωτογραφίες μετατοπισμένες μεταξύ τους ως προς τον άξονα x.

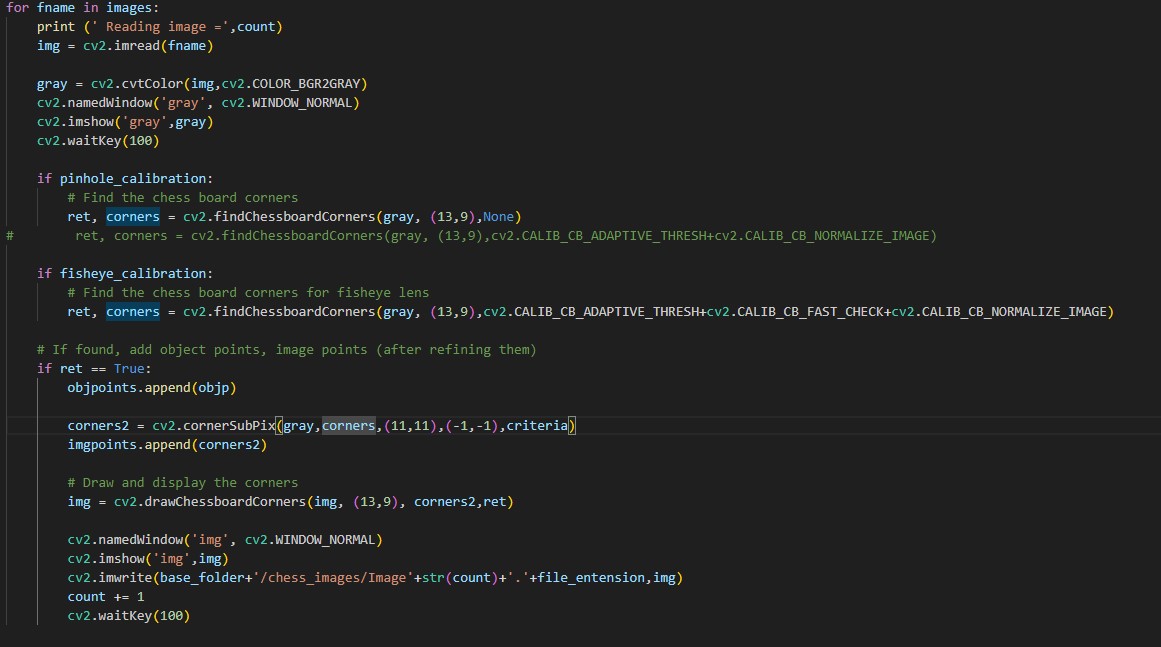
**Calibration:** Ορίζεται το μοντέλο της κάμερας. Μπορεί να είναι είτε pinhole είτε fisheye. Αυτή είναι μια επιλογή διαμόρφωσης που καθορίζει τη διαδικασία βαθμονόμησης που θα χρησιμοποιηθεί.Εμείς δουλεύουμε με τύπο pinhole, οπότε και οποιαδήποτε αναφορά στον κώδικα αφορά τις εντολές για το pinhole μοντέλο

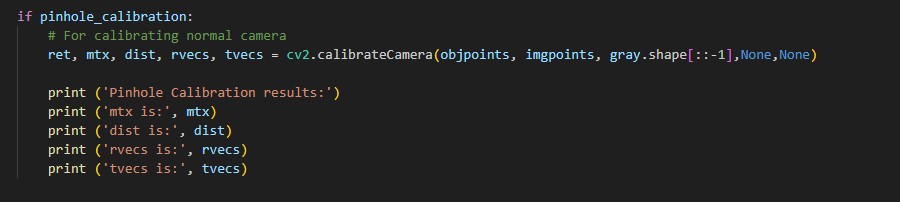
Ορισμός κριτηρίων τερματισμού για επαναληπτικούς αλγόριθμους. Σε αυτήν την περίπτωση, χρησιμοποιείται για την βελτιστοποίηση των γωνιών με τερματισμό για σφάλμα eps < 0.001 .

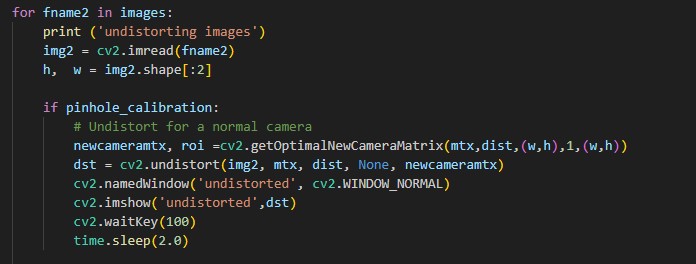
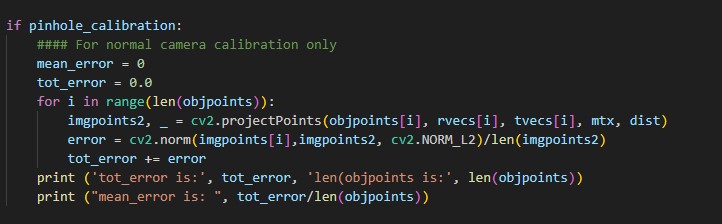


Προετοιμασία των 3D σημείων αντικειμένου (θέσεις των γωνιών στον πραγματικό κόσμο) για το πρότυπο της σκακιέρας. Αυτά τα σημεία χρησιμοποιούνται για τη βαθμονόμηση. Αρχικοποίηση λιστών για την αποθήκευση των 3D σημείων αντικειμένου και των 2D σημείων εικόνας από όλες τις εικόνες.



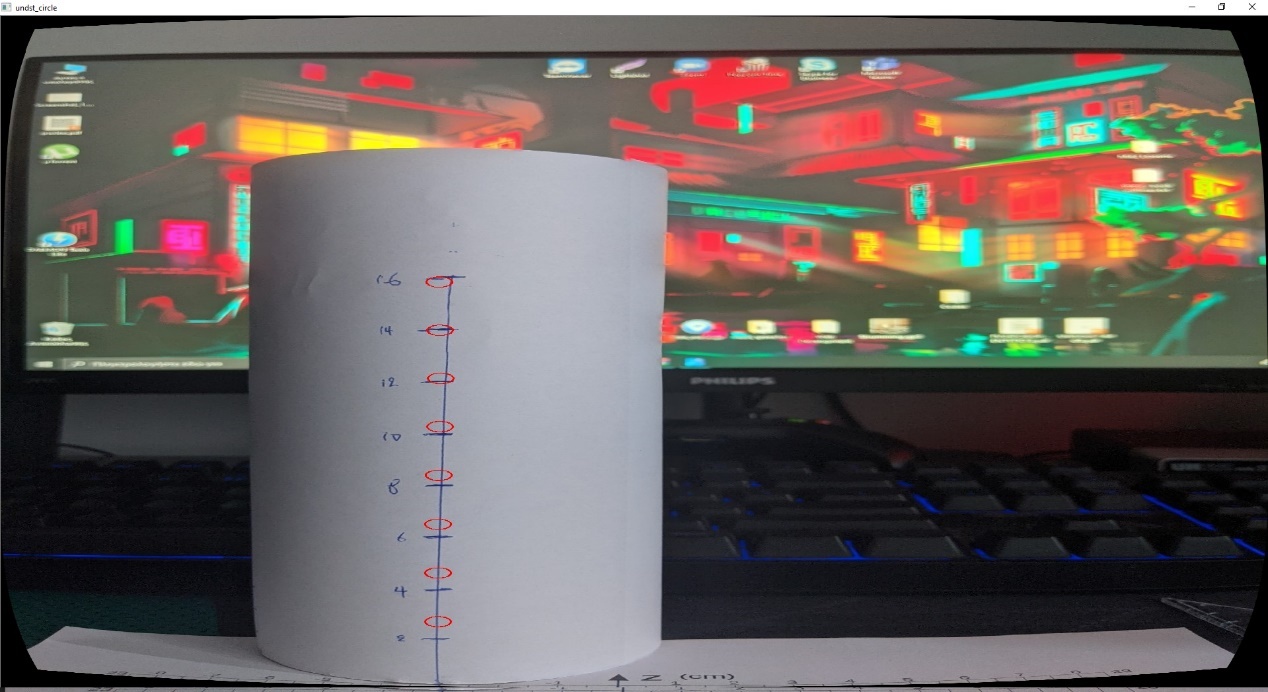
Έπειτα βρίσκονται με το glob module όλα τα αρχεία που έχουμε στο συγκεκριμένο path που έχουμε δώσει και για αυτά δημιουργούμε μια δομή επανάληψης. Κάθε εικόνα διαβάζεται σε διαβάθμιση του γκρι και με τη χρήση συνάρτησης findChessboardCorners (ορίζοντας αριθμό γραμμών και στηλών), βρίσκονται οι γωνίες της σκακιέρας. Με βάση αυτές θα βαθμονομηθεί η κάμερα. Αν έχει βρεθεί η γωνία αυξάνεται η ακρίβειά της (cornerSubPix) και με βάση τις συντεταγμένες (x,y) που λήφθηκαν, ζωγραφίζονται πάνω στην εκάστοτε εικόνα. Στο τέλος αποθηκεύουμε το αποτέλεσμα.

Με την παραδοχή ότι η σκακιέρα στα δείγματα έχει μηδενικό βάθος, λαμβάνουμε τις 3D συντεταγμένες με βάση το meshgrid που φτιάξαμε αρχικά και έχοντας πλέον τα σημεία σε 2D (image coordinates) και σε 3D (real world coordinates) καλιμπράρουμε τη κάμερα και μας επιστρέφεται ο πίνακας χαρακτηριστικών, παραμόρφωσης, θέσης και προσανατολισμού της κάμερας.

Με τους πρώτους δύο πίνακες προσομοιάζουμε μια νέα κάμερα παίρνοντας από την getOptimalNewCameraMatrix τον πίνακα βαθμονόμησης και τη περιοχή ενδιαφέροντος και διώχνουμε τη παραμόρφωση με την undistort από τις αρχικές εικόνες της σκακιέρας. Τέλος επαναπροβάλλονται τα 3D σημεία που προσεγγιστικά θεωρήσαμε ότι είναι οι γωνίες της σκακιέρας και υπολογίζεται το σφάλμα ελαχίστων τετραγώνων.

Το συνολικό σφάλμα για όλα τα σημεία είναι 1.28 pixel το οποίο σημαίνει πως η απόκλιση είναι ανεπαίσθητη. Αυτό δεν εγγυάται βέβαια ένα πετυχημένο καλιμπράρισμα, διότι μπορεί το μικρό σφάλμα να είναι απόρροια του overfitting, δηλαδή δειγμάτων-εικόνων με αρκετά παρόμοια χαρακτηριστικά, πράγμα που καθιστά ασταθή την βαθμονόμηση και την νέα κάμερα σε εικόνες με ποικίλες γωνίες, βάθος.

**Undistortion:** Σε αυτό το μέρος γνωρίζοντας τις 3D συντεταγμένες του αντικειμένου μας θέλουμε να τις μεταφέρουμε στον χώρο της οθόνης (2D). Ξεκινάμε με μια παραμορφωμένη εικόνα και όσα σημεία του χώρου θέλουμε (8 στη περίπτωσή μας) τα οποία θα αντιστοιχηθούν στο 2D παράθυρο, για να εξακριβώσουμε τoν βαθμό επιτυχίας του calibration. Όμοιες εντολές με το προηγούμενο κώδικα χρησιμοποιούνται για την διεργασία αυτή. Στο τέλος έχοντας τις 2D συντεταγμένες ζωγραφίζουμε κύκλους για να δούμε κατά πόσο επιτυχημένη ήταν η αντιστοίχιση, άρα και η βαθμονόμηση.



Τα αποτελέσματα δεν είναι ιδανικά καθώς παρουσιάζεται μια θετική απόκλιση στον άξονα y (estimation – real) όντας η κατεύθυνση του άξονα Y κατακόρυφη με φορά κάτω.

**Depth Estimation:**

**crop\_pcd**: Σχεδιάστηκε για να περικόψει ένα τρισδιάστατο σύνολο σημείων που αναπαρίσταται ως πίνακας. Έχουμε σαν όρισμα το εισερχόμενο τρισδιάστατο σύνολο σημείων pcd αναπαριστώμενο ως πίνακας NumPy, όπου κάθε γραμμή αντιστοιχεί σε ένα τρισδιάστατο σημείο, και οι στήλες αντιπροσωπεύουν τις συντεταγμένες X, Y και Z καθώς και τις ελάχιστες και μέγιστες τιμές για τον περιορισμό κατά μήκος των αξόνων X, Y και Z.

Η συνάρτηση εκτελεί τα εξής βήματα:

Δημιουργεί λογικές συνθήκες για κάθε άξονα βάσει του εάν οι τιμές των σημείων βρίσκονται εντός των καθορισμένων ορίων (x\_min έως x\_max, y\_min έως y\_max, z\_min έως z\_max).

Συνδυάζει αυτές τις συνθήκες χρησιμοποιώντας λογικές πράξεις AND για να πάρει μια συνολική συνθήκη για κάθε σημείο.

Χρησιμοποιεί το np.where για να βρει τις δείκτες των σημείων που ικανοποιούν τη συνολική συνθήκη.

Εξάγει και επιστρέφει το υποσύνολο του εισερχομένου τρισδιάστατου συνόλου σημείων που ικανοποιεί τα καθορισμένα εύρη.

**match2**: Έχει την ίδια λειτουργία με αυτή της **image\_stitching** του Α’ Μέρους

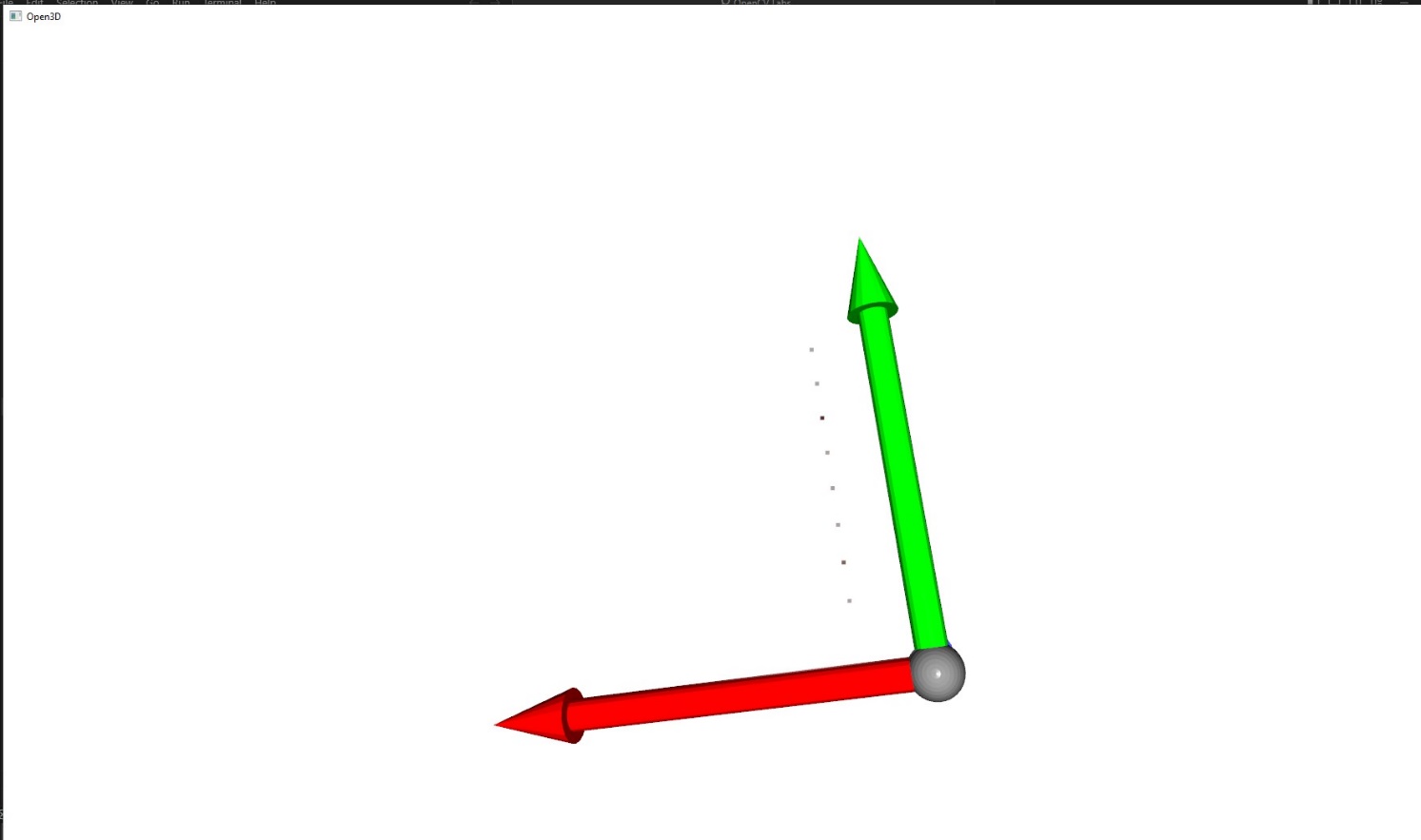
**compute\_disparity\_map:** Εκμεταλλευόμενοι δύο φωτογραφίες (τυπικά δύο κάμερες) σε απόσταση x μεταξύ τους και για τις ίδιες συντεταγμένες Y,Z μπορούμε να βρούμε το βάθος (Ζ) ενός σημείου στο χώρο (disparity=x−x′=Bf/Z).

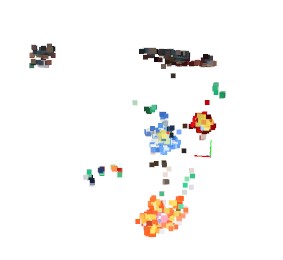
Αρχικά αφαιρούμε το μισό των διαστάσεων της εικόνας για να φέρουμε το σύστημα συντεταγμένων στο κέντρο του φακού της κάμερας. Έπειτα σύμφωνα με τον τύπο για το βάθος υπολογίζουμε το Ζ, το Χ και το Υ. Τέλος θέτουμε στο νέο τριδιάστατο σημείο το χρώμα που έχει στην αρχική εικόνα.

Στο κύριο πρόγραμμα έχουμε 2 μέρη. Στο πρώτο λαμβάνουμε τα 2Δ σημεία εικόνας και με βάση τους πίνακες που έχουμε από τη βαθμονόμηση και τη χρήση compute\_disparity\_map υπολογίζουμε τις 3D συντεταγμένες. Θέτουμε όρια έξω από τα οποία δεν λαμβάνουμε υπόψιν τα σημεία που βρήκαμε με την crop\_pcd και αναπαριστούμε αυτά που απομένουν με τη βιβλιοθήκη της open3d.

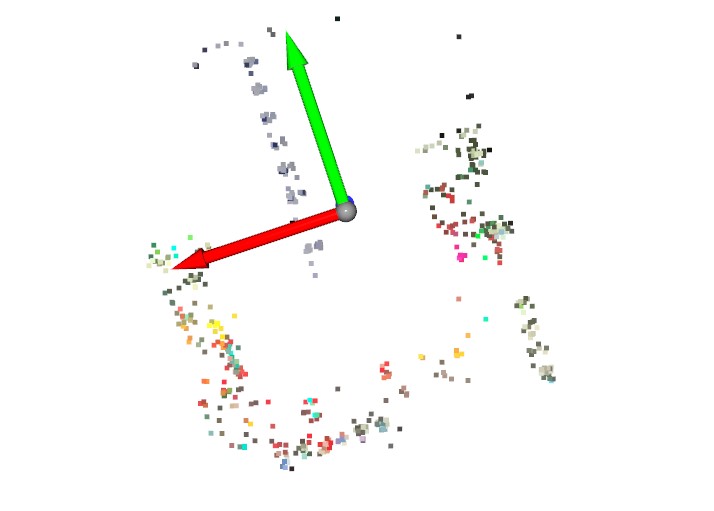
Στο δεύτερο μέρος βρίσκουμε matches μεταξύ των 2 εικόνων και κάνουμε πάλι την 3Δ ανακατασκευή τους και τη απόρριψη αυτών που είναι εκτός των ορίων x\_min, x\_max, y\_min, y\_max, z\_min, z\_max.

**Αποτελέσματα**:

****







Στην 2η 3η και 4η εικόνα, δηλαδή αυτές του 2ου μέρους του κύριου κώδικα, βλέπουμε να φαίνονται ανεστραμμένα και μετατοπισμένα τα σημεία κλειδιά ως προς το ύψος. Αυτό συμβαίνει διότι ο θετικός άξονας y έχει φορά προς τα κάτω και εμείς δεν αφαιρούμε για αυτό το 2ο μέρος του κώδικα το ύψος της κάμερας από την ανεστραμμένη συντεταγμένη Υ (-Υ δηλαδή). Αυτό φαίνεται στην γραμμή 226 και 227 και την γραμμή 85 της συνάρτησης **compute\_disparity\_map**. Ενδιαφέρουσα παρατήρηση κυρίως στη 2η εικόνα είναι πως το βάθος προκύπτει αισθητά μεγαλύτερο από αυτό που φαίνεται στην κάμερα, πράγμα λογικό αφού η κάμερα που δίνεται για αυτές τις φωτογραφίες εμφανίζει μεγάλο focal length που έχει τάση να εμφανίζει τις εικόνες πιο κοντινές, σαν να τις συμπιέζει στον άξονα του βάθους.