Ελληνικό Μεσογειακό Πανεπιστήμιο

Διατμηματικό Μεταπτυχιακό Πρόγραμμα "Προηγμένα συστήματα παραγωγής, αυτοματισμού και ρομποτικής"

Μηχανική Όραση (TM152)
Χειμερινό Εξάμηνο 2020-2021

Project 2: Features and Tracking

Ομάδα εργασίας: Ομάδα 4
Μέλη Ομάδας: Σταματάκης Δημήτριος - Ευσταθόπουλος Νικόλαος

Περιεχόμενα

Εισαγωγή για την εκτέλεση των προγραμμάτων

Στο φάκελο assignment_02_04 υπάρχει ένας φάκελος (Αναφορά) στον οποίο υπάρχει η αναφοράς (σε pdf και word), επίσης στον φάκελο assignment_02_04 υπάρχει και ένας φάκελος Αρχεία (Κώδικες κ.λπ.), στον οποίο βρίσκονται οι κώδικες για την κάθε άσκηση και επιπλέον βρίσκονται οτιδήποτε αρχεία χρειάζονται για την εκτέλεση αυτών των προγραμμάτων. Επιπλέον στον φάκελο assignment_02_04 υπάρχει ένας φάκελος Βίντεο, στον οποίο υπάρχει το τελικό βίντεο για την exercise 2 (exercise_02_video.mp4) και επιπλέον υπάρχουν τα κομμάτια αυτού του τελικού βίντεο ξεχωριστά. Τέλος, στον φάκελο assignment_02_04 υπάρχει και το pdf αρχείο με τις οδηγίες και τις εκφωνήσεις των ασκήσεων (assignment_2-FeaturesTracking.pdf).

Τα προγράμματα είναι υλοποιημένα στο περιβάλλον προγραμματισμού Spider της Anaconda και όλη η εργασία έγινε μόνο με το περιβάλλον αυτό. Πράγμα που σημαίνει ότι εάν εκτελεστούν οι κώδικες αυτοί σε άλλο περιβάλλον μπορεί να μην εμφανίζονται όλες οι εικόνες, για τον λόγω του ότι δεν υπάρχει η συνάρτηση plt.show() σε όλες τις εικόνες που εμφανίζονται. Επίσης έχει οριστεί ως φάκελος εργασίας ο φάκελος με τα αρχεία, πράγμα που σημαίνει ότι για παράδειγμα στο διάβασμα της εικόνας απλώς δίνεται το path της εικόνας από τον φάκελο εργασίας και μετά. Για ευκολότερη εκτέλεση του κώδικα, είναι καλύτερα να εκτελεστεί ο κώδικας με το περιβάλλον προγραμματισμού spyder και ορίζοντας ως φάκελο εργασίας τον φάκελο Αρχεία (Κώδικες κ.λπ.).

Κατά την εκτέλεση των προγραμμάτων (exercise_01.py και exercise_02.py) έχουν δημιουργηθεί μενού χειρισμού για ευκολότερη εκτέλεση και χειρισμού αυτών των προγραμμάτων.

Άσκηση 1: Ανίχνευση Blob

Γενικά:

Όπως είδαμε και στο κομμάτι της θεωρίας των διαλέξεων, με τον όρο "Blob" σε μια εικόνα, αναφερόμαστε σε μια ομάδα εικονοστοιχείων (συνεχόμενη περιοχή από pixels), τα οποία ικανοποιούν κάποιο κριτήριο. Στην γενική περίπτωση, αυτή η περιοχή δεν έχει απαραίτητα κάποιο συγκεκριμένο σχήμα αλλά αρκετές φορές η έννοια blob είναι συνυφασμένη με το σχήμα του κύκλου. Σαν παράδειγμα και με βάση τα όσα θα αναλυθούν παρακάτω, μπορούμε να θεωρήσουμε σαν blob το σύνολο των pixels συγκεκριμένου εύρους φωτεινοτήτων, που εσωκλείονται σε έναν κύκλο συγκεκριμένης ακτίνας.

Περιγραφή & Σκοπός

Στην άσκηση 1 κληθήκαμε σε πρώτη φάση να δημιουργήσουμε μια συνάρτηση η οποία θα είναι σε θέση να ανιχνεύει blobs σε κάποια εικόνα. Στα επόμενα ερωτήματα θα έπρεπε να εμπλουτίσουμε τη συνάρτηση αυτή με κάποιες παραμέτρους προκειμένου να κλιμακώσουμε περαιτέρω την ανίχνευση των blobs. Συγκεκριμένα, στην αρχική προσέγγιση υλοποιήσαμε την ανίχνευση ενός η περισσοτέρων blob σε μια εικόνα μιας συγκεκριμένης ακτίνας. Έπειτα εμπλουτίσαμε την προηγούμενη εκδοχή λαμβάνοντας υπόψιν ακόμα μία παράμετρο, αυτή της φωτεινότητας. Ο χρήστης μπορεί να επιλέξει αν επιθυμεί να ανιχνεύσει μια φωτεινή ή μια σκοτεινή περιοχή ή ακόμα και τις δυο, αλλά πάντα συγκεκριμένης ακτίνας. Τέλος , κλιμακώνοντας περαιτέρω το πρόβλημα, εμπλουτίσαμε την προηγούμενη εκδοχή, εντοπίζοντας blobs σε μια εικόνα παραμετροποιημένα ως προς τη φωτεινότητα και το εύρος

της ακτίνας. Στην προκείμενη περίπτωση ο χρήστης θα μπορεί να εισάγει ένα εύρος τιμών ακτίνων (rmin,rmax) και να εντοπίζει blobs σε μια εικόνα που θα κυμαίνονται σε αυτό το εύρος. Όλη η παραπάνω διαδικασία έχει υλοποιηθεί με τη μορφή menu. Συνίσταστε η συγκεκριμένη σειρά που προαναφέρθηκε για την καλύτερη κατανόηση του blob detection, χωρίς αυτό να είναι δεσμευτικό. Δηλαδή ο χρήστης αν το επιθυμεί, μπορεί κάλλιστα να επιλέξει να δει εξ' αρχής blobs κλιμακούμενης ακτίνας και διάφορων φωτεινοτήτων.

Υλοποίηση Άσκησης 1

Αρχικά Menu επιλογών

Στο σημείο αυτό θα αναφερθούν αναλυτικά τα βήματα που ακολουθήθηκαν για την επίλυση της πρώτης άσκησης. Συγκεκριμένα μόλις ο χρήστης εκτελέσει το πρόγραμμα θα δει τα παρακάτω menu: **(βήμα 1)**

Menu 1

1) black_dots
2) white_dots
3) coins
4) circles_{}
5) Allo path
6) Allo path apo diadyktio
7) Exit

Ο χρήστης καλείται να επιλέξει σε ποια απ' τις εικόνες που μας δόθηκαν και όχι μόνο, θέλει να ανιχνεύσει blobs

Menu 2

1) Sygkekrimeni timi aktinas
2) Eyros aktinas

Συνεπώς, η δεύτερη επιλογή που καλείται να κάνει είναι σχετικά με το αν θέλει να δει blobs συγκεκριμένης ακτίνας ή εύρους ακτίνων.

Menou 3

1) Dark blobs?
2) Light blobs?
3) Both?

Τέλος ο χρήστης καλείται να πάρει απόφαση σχετικά με τη φωτεινότητα(/ες), των blobs που θέλει να δει.

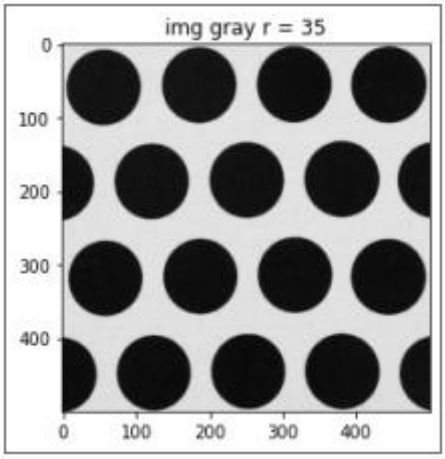
Έχοντας πάρει τις παραπάνω αρχικές αποφάσεις, τρέχουν οι εκαστοτε αλγόριθμοι εσωτερικά στον κώδικα και εμφανίζονται στο χρήστη η εικόνα με τα blobs που έχει επιλέξει.

Αλγοριθμική Προσέγγιση Άσκησης 1

Εντοπισμός Blobs δεδομένης ακτίνας και φωτεινότητας

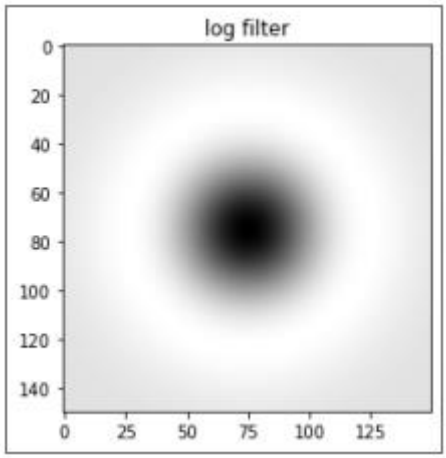
Εφόσον ο χρήστης έχει επιλέξει απ΄ τα menu το συγκεκριμένο συνδυασμό τότε ακολουθείται η εξής αλγοριθμική διαδικασία για την τελική εύρεση blob:

Αρχικά η εικόνα που έχει επιλέξει μετατρέπεται σε grayscale μέσω της συνάρτησης cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)) για απλότητα των υπολογισμών. Έστω ότι επέλεξε την εικόνα black_dots.jpg απ το dataset: **(βήμα 2)**



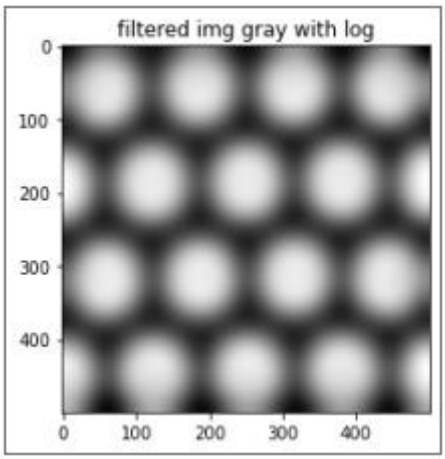
black_dots

Στη συνέχεια φιλτράρουμε την εικόνα με τη Λαπλασιανή της Γκαουσιανής (LoG) μέσω της συνάρτησης που μας δόθηκε, logkern(sigma). Το sigma της Λαπλασιανή της Γκαουσιανής, όπως έχει αναλυθεί και στις διαφάνειες **sigma=radius/np.sqrt(2.0).** Το LoG filter αυτού του παραδείγματος φαίνεται παρακάτω. **(βήμα 3)**



log filter

Η εικόνα μετά την εφαρμογή του LoG filter είναι η εξής:

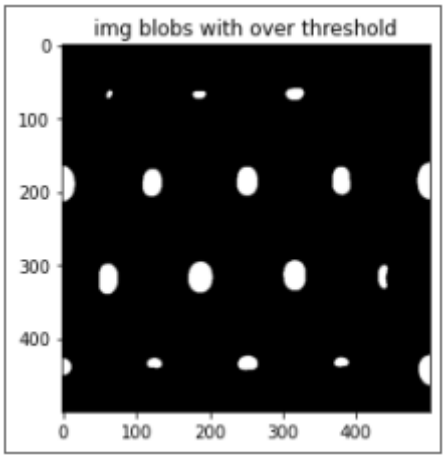


filtered image

Μετά εφαρμόζουμε μια μάσκα στην εικόνα, βάσει του αποτελέσματος της οποίας θα εξάγουμε μετέπειτα τα dark και τα light blobs. Συγκεκριμένα η **img_log[img_log < 0] = 0** χρησιμοποιείται στην περίπτωση που θέλουμε να εξάγουμε μετέπειτα τα dark blobs και η **img_log[img_log > 0] = 0** για τα light blobs. **(βήμα 4)**

Έπειτα κατωφλοιώνουμε το εκάστοτε αποτέλεσμα ρυθμίζοντας έτσι και την ένταση της επιλογής που έχει κάνει ο χρήστης, δηλαδή το κατά πόσο "dark" η "light" θέλουμε να είναι το περιεχόμενο των blobs. Ως thres_ratio έχει δοθεί by default το 0.7. **(βήμα 5)**

thres = thres_ratio * max_val
img_blobs = np.abs(img_log) > thres



thresholted image

Στη συνέχεια καλούμε τη συνάρτηση **detect_blobs()** με τα κατάλληλα ορίσματα σύμφωνα με τις επιλογές που έχει κάνει ο χρήστης απ τα menu για εύρεση blobs. Βρίσκουμε τα μέγιστα σημεία των περιοχών αυτών τα οποία εξ' ορισμού απ' την εφαρμογή του LoG filter και όπως έχουμε αναφέρει και στις διαλέξεις, είναι τα κέντρα των κύκλων (blob).Η εύρεση των σημείων αυτών γίνεται με τη βοήθεια της συνάρτησης που μας δόθηκε **local_maxima_3D,** η οποία επιστρέφει μια λίστα με ζεύγη (tuples), που έχουν την ακτίνα και τη θέση x,y του κέντρου του εκάστοτε blob, το οποίο πληρεί τα κριτήρια που αναζητά ο χρήστης στην εικόνα. Προφανώς στην περίπτωση που δουλεύουμε με σταθερές ακτίνες, το δεύτερο index απ' τα tuples της λίστας που επιστρέφει η local_maxima_3D, θα είναι σταθερό και ίσο με την ακτίνα που θα δώσει ο χρήστης και το πρώτο θα περιέχει τις θέσεις των blobs με τη συγκεκριμένη ακτίνα. **(βήμα 6).**

Στη συνέχεια δημιουργούμε ένα αντίγραφο της εικόνας np.zeros_like(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)),δηλαδή ένα μαύρο καμβά και στις συντεταγμένες των κέντρων των blobs που βρέθηκαν στο προηγούμενο βήμα δίνουμε την τιμή της ακτίνας. **(βήμα 7)**

Τέλος σχεδιάζουμε τα blobs στην αρχική εικόνα χρησιμοποιώντας τον αντίστοιχο κώδικα που μας δόθηκε στο tutorial_5. **(βήμα 8)**

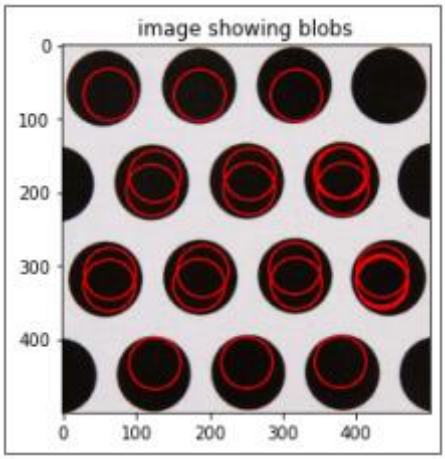
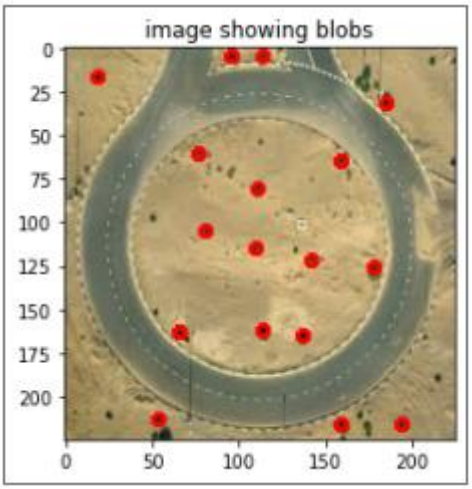


Image with dark blobs r=35

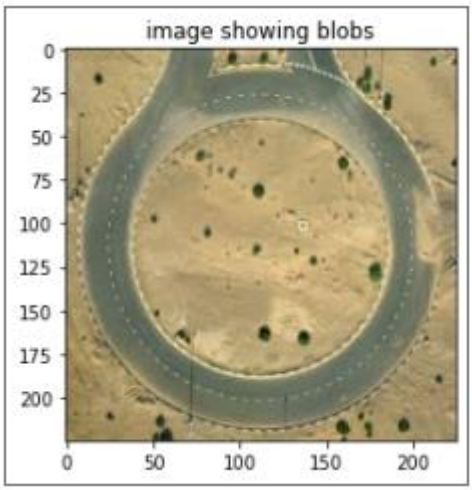
<u>Αποτελέσματα</u>

Ακολουθεί μια γενική παράθεση αποτελεσμάτων της εκτέλεσης του προγράμματος. Ως thres_ratio έχει θεωρηθεί το 0.6

Εικόνα : circles_03.jpg. Αναζήτηση dark blobs

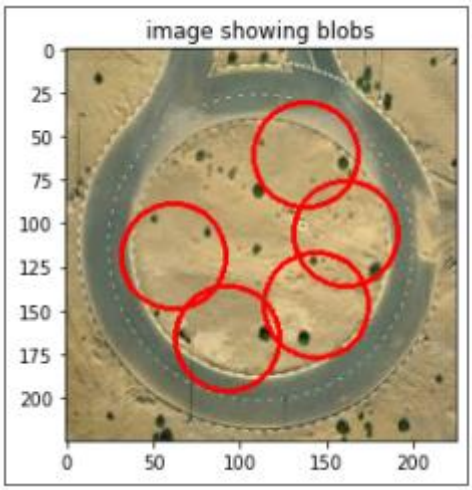


Ακτίνα αναζήτησης 3

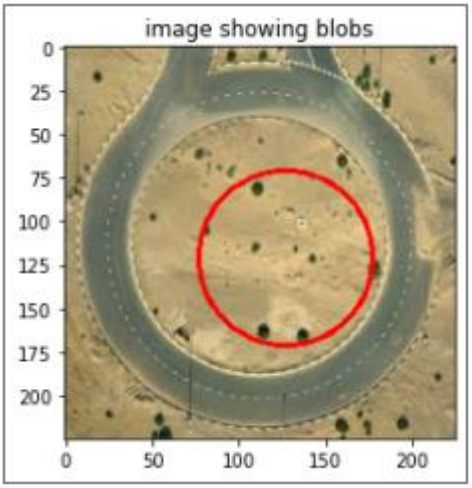


Ακτίνα αναζήτησης 10

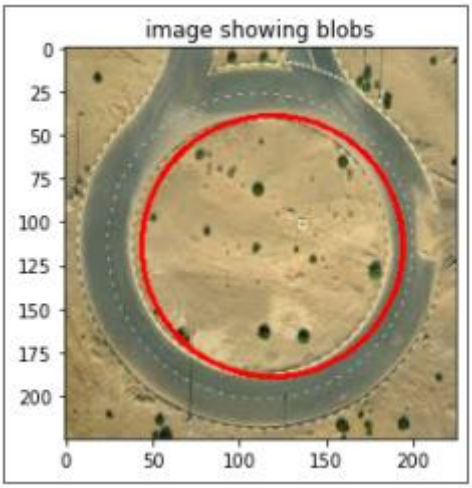
Εικόνα : circles_03.jpg. Αναζήτηση light blobs



Ακτίνα αναζήτησης 30

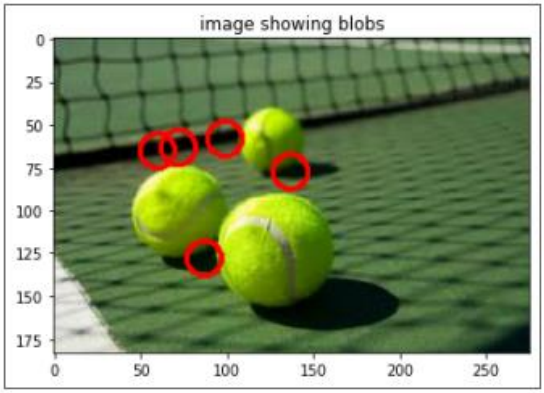


Ακτίνα αναζήτησης 50

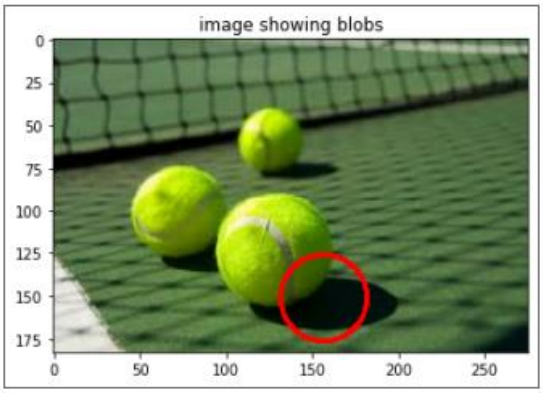


Ακτίνα αναζήτησης 75

Εικόνα : circles_04.jpg. Αναζήτηση dark blobs

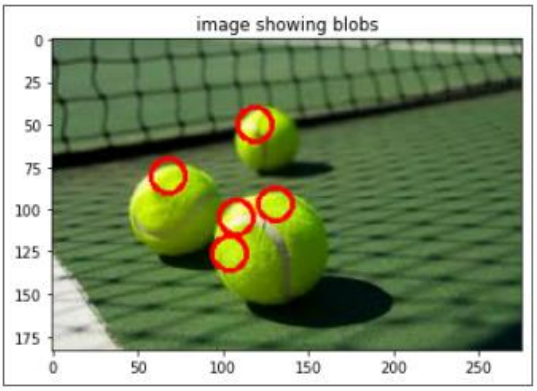


Ακτίνα αναζήτησης 10

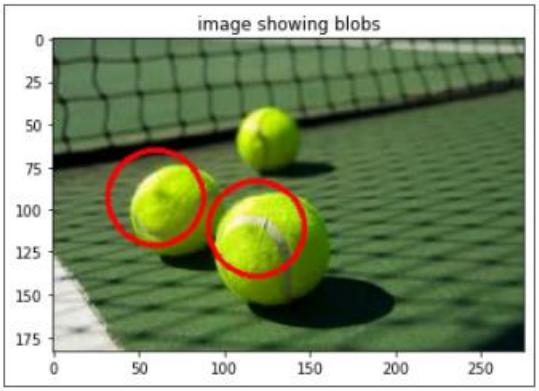


Ακτίνα αναζήτησης 25

Εικόνα : circles_04.jpg. Αναζήτηση light blobs

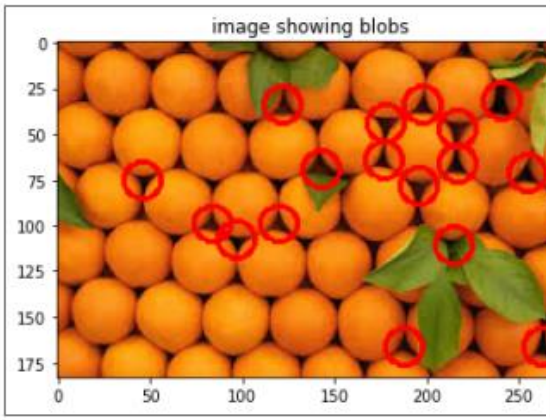


Ακτίνα αναζήτησης 10

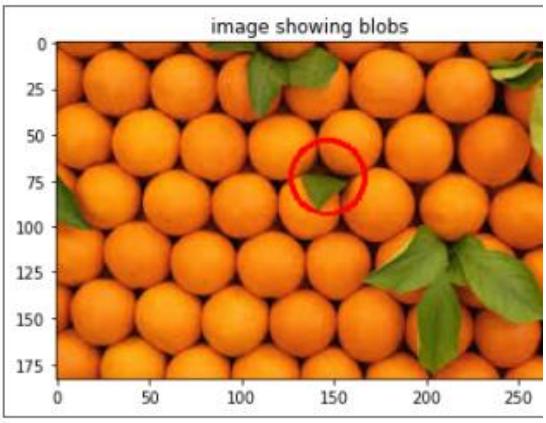


Ακτίνα αναζήτησης 28

Εικόνα : circles_05.jpg Αναζήτηση dark blobs

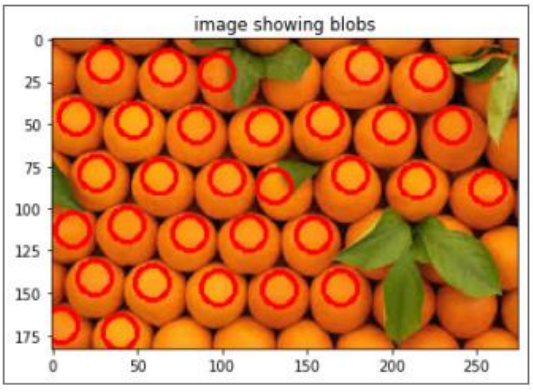


Ακτίνα αναζήτησης 10

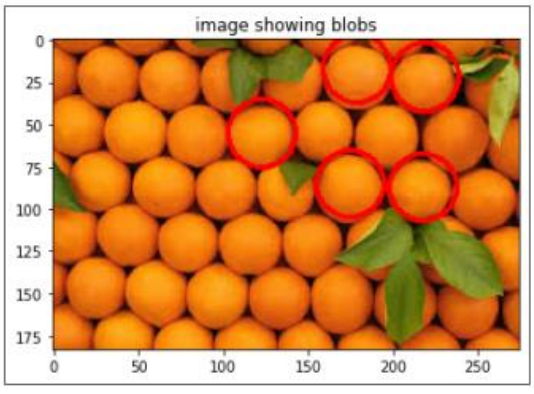


Ακτίνα αναζήτησης 20

Εικόνα : circles_05.jpg. Αναζήτηση light blobs



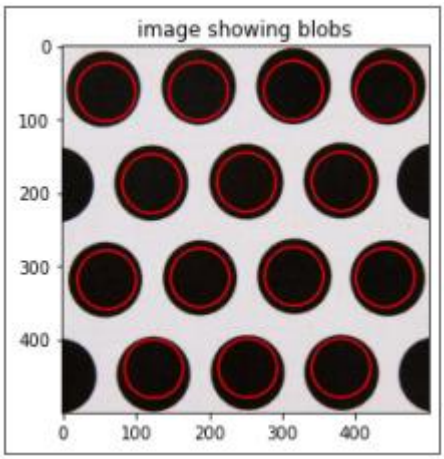
Ακτίνα αναζήτησης 10



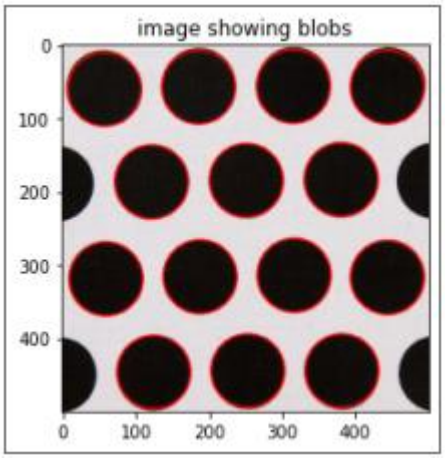
Ακτίνα αναζήτησης 20

<u>Παρατηρήσεις βάσει των αποτελεσμάτων</u>

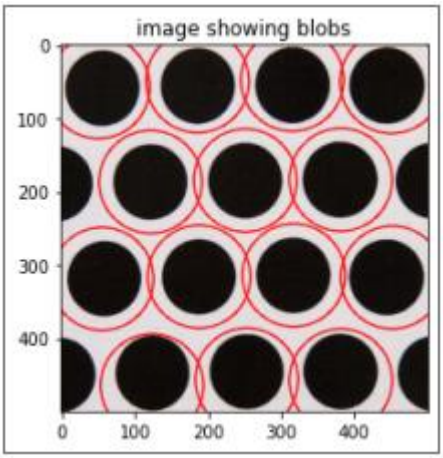
Σε αυτό το σημείο ρίχνουμε μια πιο αναλυτική ματιά στα αποτελέσματα της εκτέλεσης του αλγορίθμου για διαφορετικές τιμές ακτίνας πχ στην εικόνα black_dots.jpg



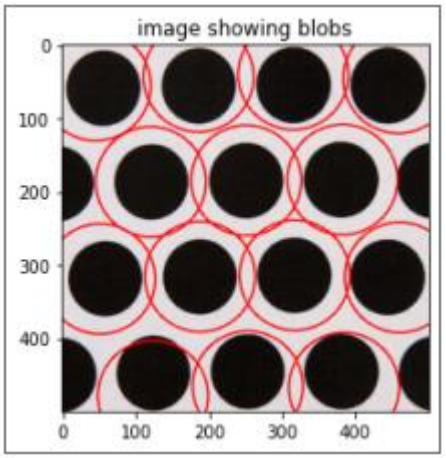
Ακτίνα αναζήτησης 40



Ακτίνα αναζήτησης 50



Ακτίνα αναζήτησης 70

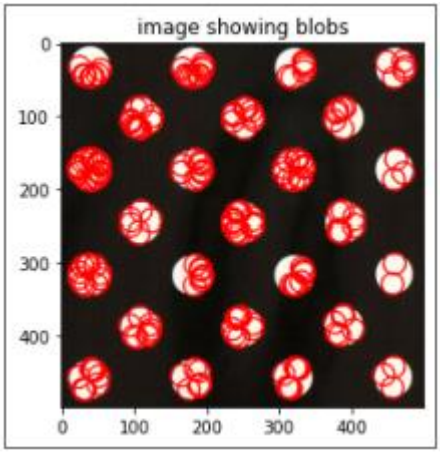


Ακτίνα αναζήτησης 75

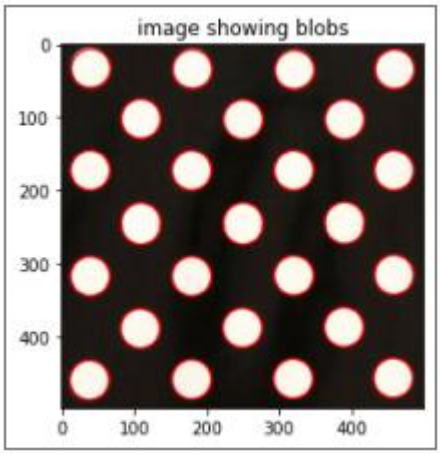
Παρατηρήσεις

Από μέτρηση που έχει προηγηθεί (χοντρικά), ο μαύρος κύκλος έχει μέγεθος 50 pixels. Οπότε αν ζητήσω να εντοπιστεί μια σκούρα περιοχή αυτού του μεγέθους τότε η κατωφλίωση μου είναι εξαιρετικά επιτυχημένη και γενικά έχουν εντοπιστεί σωστά τα dark blobs.

Παρακάτω δίδονται τα αποτελέσματα της εκτέλεσης του αλγορίθμου για διαφορετικές τιμές ακτίνας στην εικόνα white_dots.jpg.



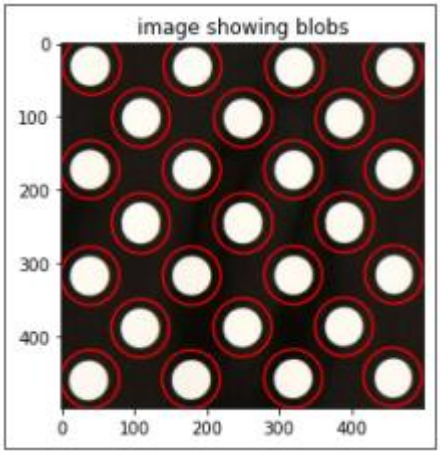
Ακτίνα αναζήτησης 15



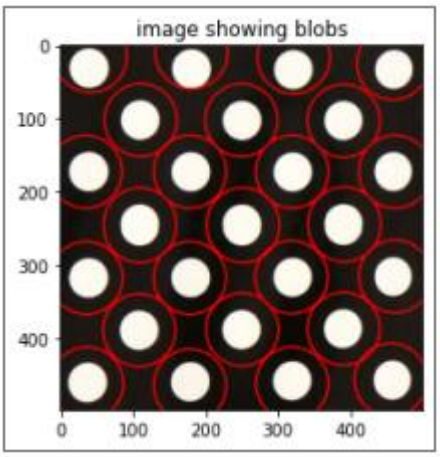
Ακτίνα αναζήτησης 27

Παρατηρήσεις

Από μέτρηση που έχει προηγηθεί (χοντρικά), ο λευκός κύκλος έχει μέγεθος κάτι λιγότερο από 30 pixels. Οπότε αν ζητήσω να εντοπιστεί μια ανοιχτόχρωμη περιοχή αυτού του μεγέθους τότε η κατωφλίωση μου είναι αρκετά επιτυχημένη γενικά. Βέβαια είναι φανερό ότι όσο μεγαλύτερη περιοχή ζητάω τόσο περισσότερα θα εκτείνεται η περιοχή από το κέντρο των εκάστοτε blobs και τόσο θα μεγαλώνει το σφάλμα μου.



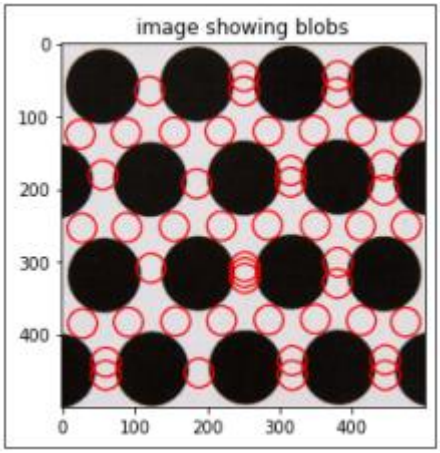
Ακτίνα αναζήτησης 40



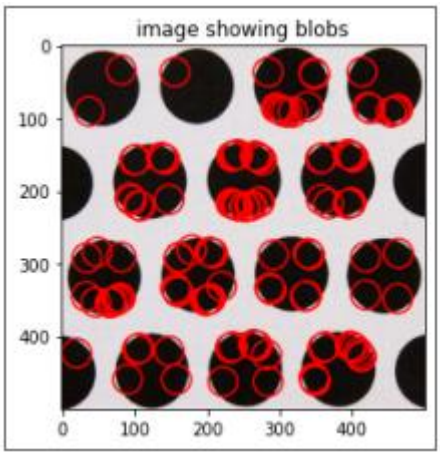
Ακτίνα αναζήτησης 50

Παρατηρήσεις

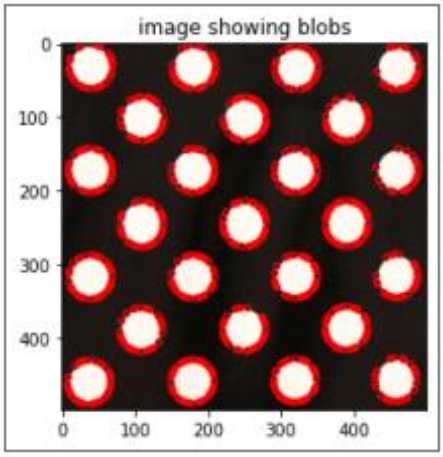
Επιπλέον αξίζει να αναφερθεί ότι αν ο χρήστης ζητήσει στην εικόνα white_dots σκούρα blobs παίρνει σαν αποτέλεσμα περιοχές που έχουν στην πλειοψηφία τους σκούρα πληροφορία και περιορισμένη (έως και καθόλου) λεύκη. Επίσης αν ζητήσει ανοικτές περιοχές σε εικόνες που έχουν πληροφορία σε λευκή περιοχή, όπως η white_dots, το αποτέλεσμα είναι να παίρνει περιοχές που έχουν στην πλειοψηφία τους ανοιχτόχρωμη πληροφορία και περιορισμένη (έως και καθόλου) σκουρόχρωμη. Προφανώς τα ακριβώς αντίθετα συμπεράσματα παρατηρούνται στην black_dots. Παρακάτω ακολουθούν εικόνες που αποδεικνύουν όσα προαναφέρθηκαν.



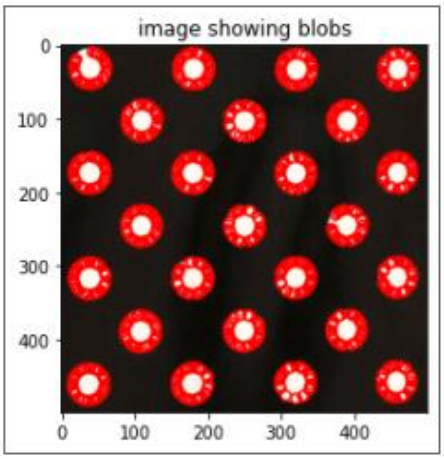
light blobs ακτίνας 20 στην black_dots.jpg



dark blobs ακτίνας 20 στην black_dots.jpg



dark blobs ακτίνας 5 στην white_dots.jpg



light blobs ακτίνας 5 στην white_dots.jpg

Τέλος αξίζει να τονιστεί το γεγονός ότι το πόσες και ποιες περιοχές θα ανιχνεύσουμε τελικά, σχετίζεται άμεσα και με το threshold. Αν μπουν πολύ "αυστηρά" threshold πχ 0.8 ή 0.9 σε "ισορροπημένες" εικόνες στο σύνολο τους, που δεν έχουν έντονα σκούρα ή έντονα λευκά τότε ο blob detecor για κάποιες ακτίνες μπορεί και να μην επιστρέψει τίποτα.

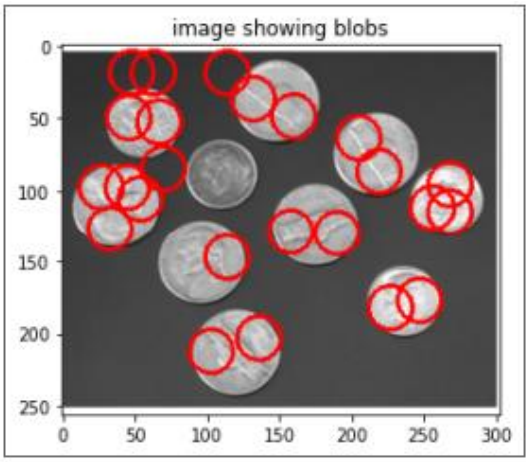
Όλες οι προηγούμενες παρατηρήσεις, αλλά ακόμα περισσότερο οι δυο τελευταίες ισχυροποιούν την υλοποίηση μας σχετικά με την ανίχνευση των blobs.

Εντοπισμός Blobs δεδομένης ακτίνας και μεικτών φωτεινοτήτων

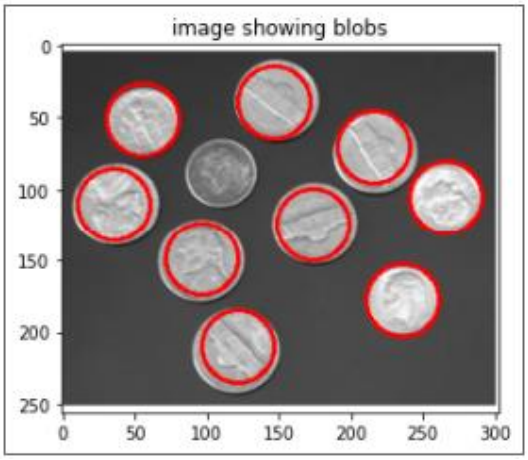
Ουσιαστικά σ' αυτή την περίπτωση ακολουθούνται τα ίδια βήματα στην εύρεση των blobs με αυτά που αναφέρθηκαν προηγουμένως. Η βασική διαφορά είναι ότι δεν εφαρμόζεται κάποια μάσκα η οποία "ξεσκαρτάρει" μόνο τις σκουρόχρωμες ή μόνο τις ανοιχτόχρωμες περιοχές της εικόνας, καθώς σε αυτή την περίπτωση τις θέλουμε όλες.

Αποτελέσματα & Παρατηρήσεις

Εικόνα coins.jpg. Αναζήτηση μεικτό φωτισμό

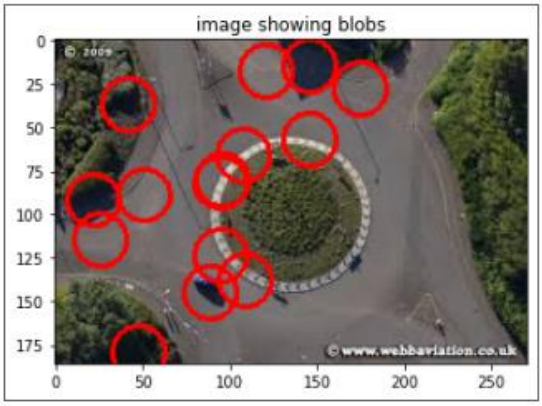


Ακτίνα αναζήτησης 15

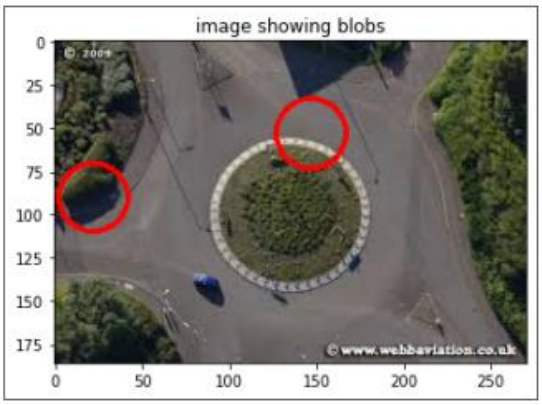


Ακτίνα αναζήτησης 25

Εικόνα circles_01.jpg. Αναζήτηση μεικτό φωτισμό

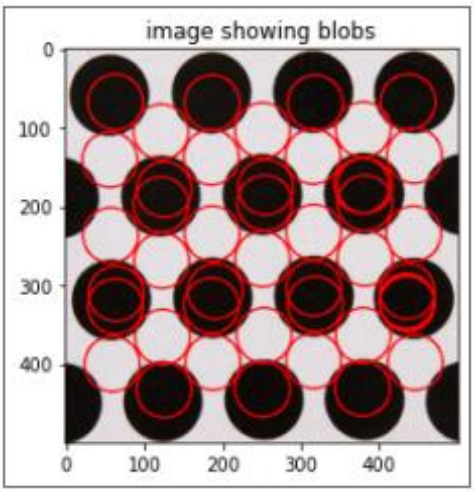


Ακτίνα αναζήτησης 15

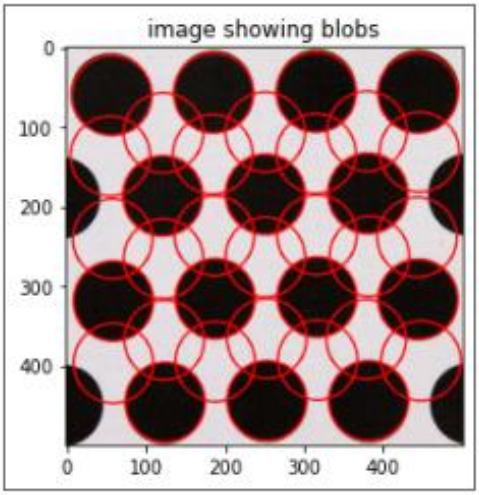


Ακτίνα αναζήτησης 20

Εικόνα black_dots.jpg. Αναζήτηση μεικτό φωτισμό

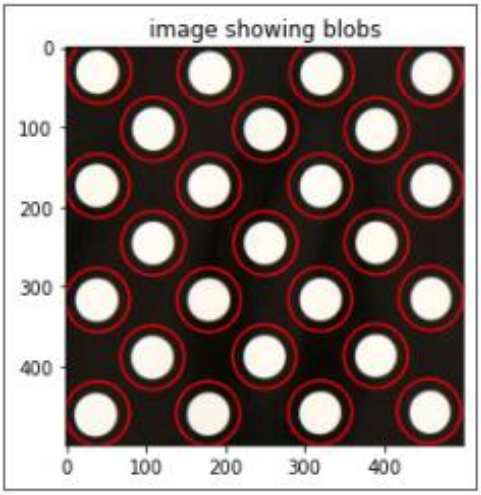


Ακτίνα αναζήτησης 35

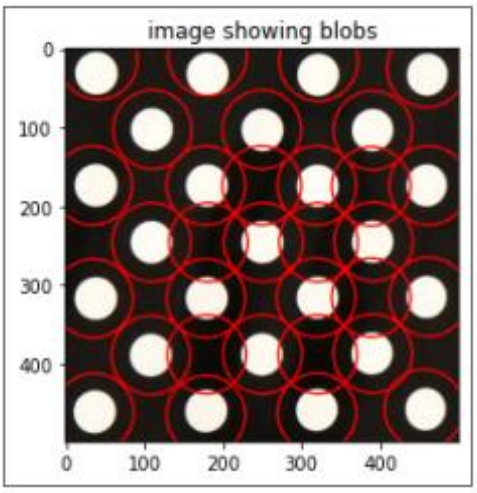


Ακτίνα αναζήτησης 50

Εικόνα white_dots.jpg. Αναζήτηση μεικτό φωτισμό



Ακτίνα αναζήτησης 35



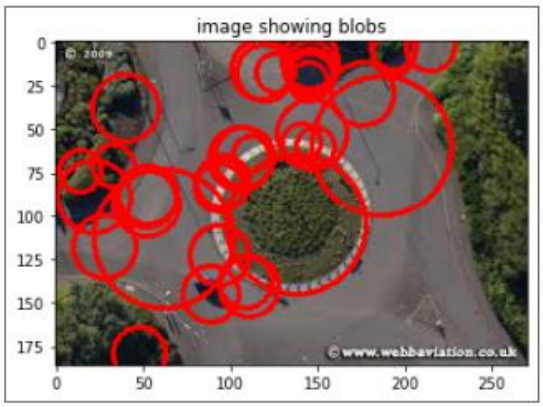
Ακτίνα αναζήτησης 50

Συνεπώς, παρατηρούμε ότι ο αλγόριθμος λειτουργεί αρκετά καλά για αναζητήσεις είτε σκούρων, είτε ανοιχτόχρωμων, είτε περιοχών με συνδυασμό βάρους φωτεινότητας για την εκάστοτε τιμή ακτίνας που επιλέγει ο χρήστης.

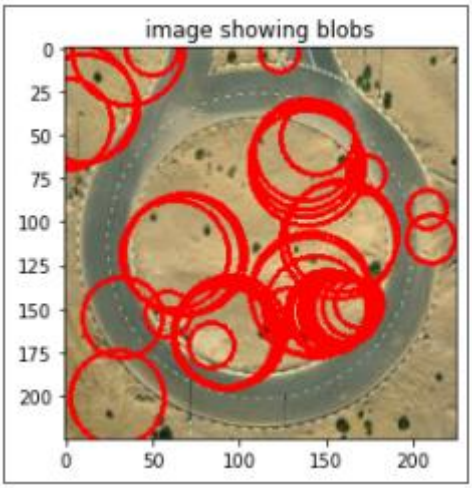
Εντοπισμός Blobs εύρους ακτίνων

Τα βήματα για την επίλυση αυτού του ερωτήματος είναι παρόμοια με αυτά που προαναφέρθηκαν με κάποιες επιμέρους μικρές διαφορές. Η βασική διαφορά είναι στα βήματα 6 και 7. Όσο αφορά το βήμα 6, προφανώς στην περίπτωση που δουλεύουμε με εύρος ακτίνων, το πρώτο index απ' τα tuples της λίστας που επιστρέφει η local_maxima_3D, θα περιέχει όλες τις ακτίνες για τις οποίες βρέθηκαν blobs που ικανοποιούν τις προϋποθέσεις έντασης φωτεινότητας, τις οποίες ζήτησε ο χρήστης. Στο δεύτερο index θα υπάρχει ο πίνακας με τις θέσεις των ακτίνων. Αντίστοιχα στο βήμα 7, όπου μαρκάρουμε τις περιοχές, στην εκαστοτε θέση που βρέθηκε το blob θα μένει κάθε φορά η μεγαλύτερη ακτίνα όπως είναι υλοποιημένος ο κώδικας. Για παράδειγμα σε περίπτωση που στη θέση (200,200) υπάρχουν 2 υποψήφια blobs, τότε θα κρατήσουμε αυτό με την μεγαλύτερη ακτίνα.

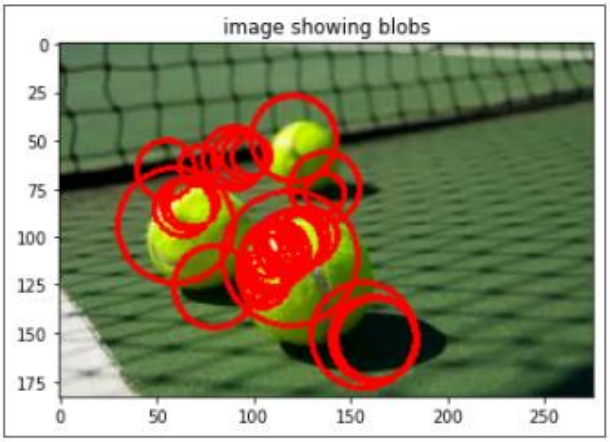
Αποτελέσματα & Παρατηρήσεις



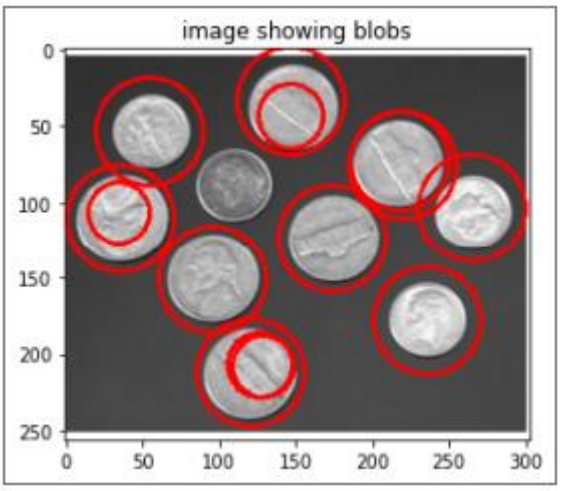
Εικόνα circles_01.jpg. Αναζήτηση μεικτό φωτισμό για ακτίνες 10-40



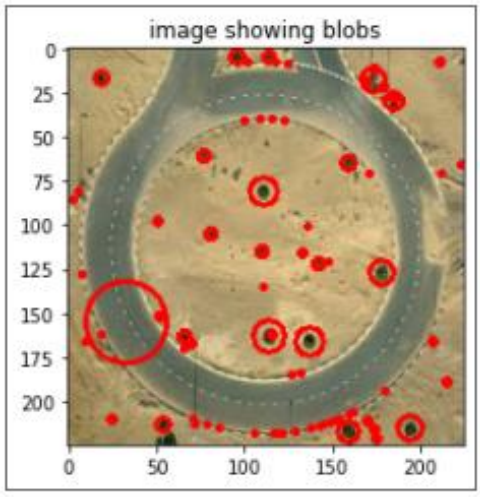
Εικόνα circles_03.jpg. Αναζήτηση μεικτό φωτισμό για ακτίνες 10-35



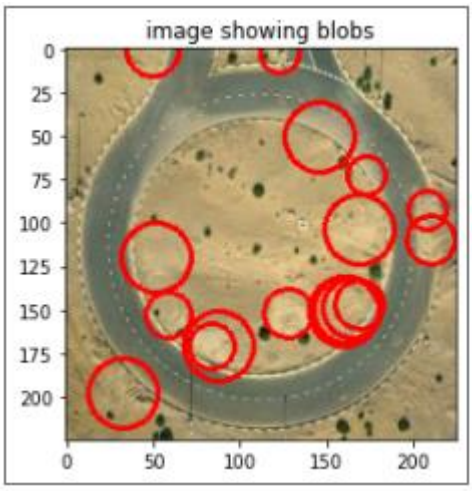
Εικόνα circles_04.jpg. Αναζήτηση μεικτό φωτισμό για ακτίνες 10-35



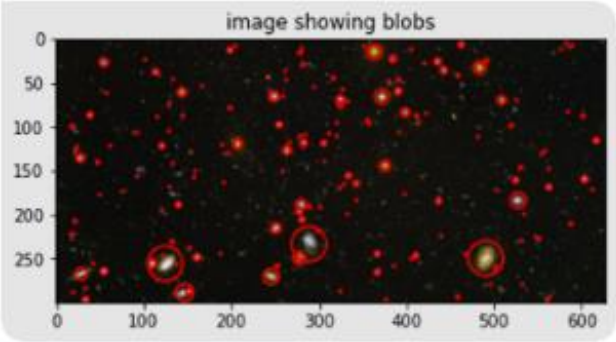
Εικόνα coins.tiff. Αναζήτηση μεικτό φωτισμό για ακτίνες 20-35



Εικόνα circles_03.jpg. Αναζήτηση dark_blobs για ακτίνες 1-35



Εικόνα circles_03.jpg. Αναζήτηση light_blobs για ακτίνες 10-20



Εικόνα από tutorial 5. Αναζήτηση light blobs για ακτίνες 1-20

Άσκηση 2

Εισαγωγή

Η άσκηση αυτή είναι υλοποιημένη για σύγκριση template με εικόνες όπου το αντικείμενο που εμφανίζεται στο template εμφανίζεται με ικανοποιητικό μέγεθος στην εικόνα. Δηλαδή συγκρίνεται εν ολίγης το template με την εικόνα. Άρα βρίσκονται features στο template και σε ολόκληρη την εικόνα, τα οποία στην συνέχεια αντιστοιχούνται. Αυτό δουλεύει ικανοποιητικά με τα αρχεία του melita, αλλά δεν δουλεύει ικανοποιητικά στα υπόλοιπα αρχεία.

Για να δουλεύει ικανοποιητικά σε όλα τα αρχεία, πρέπει να επεκταθεί και να επεξεργαστεί ο αλγόριθμος που υλοποιήθηκε, ώστε το template να υπάρχει για διάφορα scale και στην συνέχεια να συγκρίνεται το template με διάφορα scale με κάποια εικόνα, ώστε να εντοπιστεί το καλύτερο σημείο που το αντιστοιχεί, το οποίο στην ουσία θα είναι ένα σημείο που βρίσκεται στην εικόνα και στην συνέχεια με κατάλληλα βήματα να σχεδιάσουμε το πλαίσιο αυτό στην εικόνα μας και να δουλέψουμε στην συνέχεια με αυτό το πλαίσιο.

Αν επεκταθεί ο κώδικας – αλγόριθμος σύμφωνα με την ιδέα που αναφέρθηκε προηγουμένως, τότε απλώς θα γίνεται εντοπισμός χαρακτηριστικών (features) στο template και στο πλαίσιο εκείνο της εικόνας (Δηλαδή δεν θα γίνει εντοπισμός features σε όλη την εικόνα, αλλά σε ένα εύρος pixels που θα απεικονίζουν το template μας). Στην συνέχεια έχοντας βρει τα features στο template και στην εικόνα, θα γίνει η αντιστοίχιση αυτών και θα δουλεύει αρκετά ικανοποιητικά.

Εάν δεν γίνει αυτό που αναφέρθηκε προηγουμένως, σε περίπτωση που θελήσουμε να αντιστοιχίσουμε features του template με features της εικόνας, τότε θα γίνει εντοπισμός features στο template και σε ολόκληρη την εικόνα! Πράγμα που σημαίνει ότι θα γίνει αντιστοίχιση των features του template με τα features ολόκληρης της εικόνας, όπου στην περίπτωση αυτή μπορεί να υπάρχουν αντιστοιχίες features (πιο σωστά keypoints) του template με keypoints της εικόνας σε σημεία που δεν θέλουμε (δηλαδή κάποιο keypoint του template να αντιστοιχεί με ένα keypoint της εικόνας που βρίσκεται μακριά από το αντικείμενο που θέλουμε να συγκρίνουμε), διότι ο περιγραφέας (descriptor) είναι αρκετά ίδιος.

Επομένως όπως αναφέρθηκε και στην αρχή, η άσκηση αυτή δεν έχει επεκταθεί βάση αυτών που ανφέρθηκαν προηγουμένως, πράγμα που σημαίνει ότι κάθε φορά συγκρίνεται το template μας με ολόκληρη την εικόνα. Για το παράδειγμα του Melita δουλεύει ικανοποιητικά.

Επίλυση της άσκησης

Για την επίλυση της άσκησης έχουν χρησιμοποιηθεί αλγόριθμοι από τα tutorials του εργαστηριακού μέρους του μαθήματος, με την κατάλληλη επεξεργασία όπου χρειαζόταν. Επίσης έχουν υλοποιηθεί και επιπλέον αλγόριθμοι – συναρτήσεις για την επίλυση της άσκησης.

Επεξήγηση του κώδικα και των αλγορίθμων μαζί και με παράδειγμα

Στις πρώτες γραμμές έχουν συμπεριληφθεί όλες οι βιβλιοθήκες που μας χρειάστηκαν. Έπειτα ακολουθούν οι συναρτήσεις που υλοποιήθηκαν και χρησιμοποιήθηκαν για την υλοποίηση αυτή της άσκησης και τέλος υπάρχει το κύριο πρόγραμμα.

Στο κύριο πρόγραμμα έχει χρησιμοποιηθεί αρχικά το διάβασμα των εικόνων όπως υπήρχε στα tutorials, για λόγους ευκολίας και εξοικονόμησης χρόνου. Ξεκινάμε και λαμβάνουμε το

template με παρόμοιο τρόπο όπως στα tutorials, το οποίο και σχεδιάζεται πάνω στην εικόνα που έχει ληφθεί και εμφανίζεται και ξεχωριστά.

Στην συνέχεια ακολουθεί μία δομή επανάληψης, όπου για κάθε επανάληψη θα γίνεται αντιστοίχιση των features του template με τα features της εικόνας, μερικά από τα οποία θα εμφανίζονται σε εικόνα και στην συνέχεια ακολουθεί η εύρεση του affine μετασχηματισμού χωρίς την χρήση του αλγορίθμου RANSAC, η εύρεση του affine μετασχηματισμού με την χρήση του αλγορίθμου RANSAC, η εύρεση τους μετασχηματισμού homography χωρίς την χρήση του αλγορίθμου RANSAC και η εύρεση του μετασχηματισμού homography με την χρήση του αλγορίθμου RANSAC. Μόλις έχουν βρεθεί οι μετασχηματισμοί που αναφέρθηκαν προηγουμένως, περνάμε στον μετασχηματισμό των σημείων του template σε σημεία πάνω στην εικόνα και γίνεται η απεικόνιση των σημείων αυτών με σχεδιασμένες ευθείες, από το ένα σημείο στο άλλο, προσπαθώντας δηλαδή να σχεδιάσουμε τις διαστάσεις του αντικειμένου (που μοιάζει με το template) στην εικόνα.

Ας περάσουμε σε μερικά παραδείγματα για να δούμε μαζί αυτά που περιγράφηκαν προηγουμένως στην πράξη.

Δημιουργία Template

Ξεκινάμε με την δημιουργία του template:

Αρχικά διαβάζουμε μια εικόνα από την οποία λαμβάνουμε το template, η απόκτηση του template βασίζεται στα σημεία που μας δίνονται μέσω αρχείου .txt που βρίσκεται στον φάκελο με τα αρχεία του Melita. Στην περίπτωση αυτή έχει χρησιμοποιηθεί η εικόνα 0, διότι έχει καλύτερα σημεία για template.

Εικόνα 1

Εικόνα 2

Εικόνα 3

```
TrackingDatasets/Melita/img/00000.png
((128, 255), (298, 498))
TrackingDatasets/Melita/img/00000.png
[[  0 170 170   0]
 [  0   0 243 243]]
```

Εικόνα 4

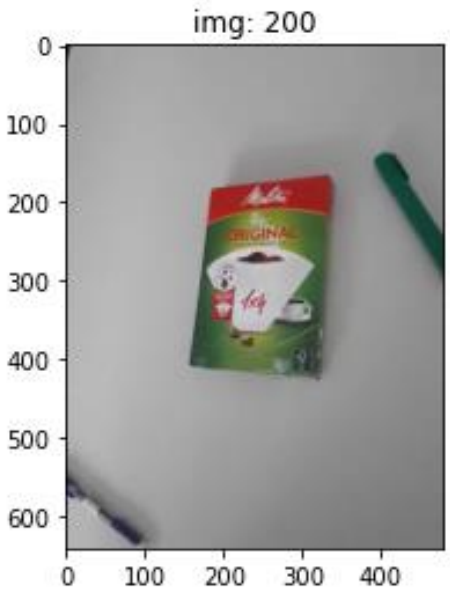
Στην **Εικόνα 4** φαίνεται εν ολίγης αρχικά το πάνω αριστερό σημείο του template και το κάτω δεξί σημείου του template στην εικόνα όπου έχει ληφθεί και στην συνέχεια φαίνονται τα άκρα του template (σημεία άκρων της template εικόνας).

Αφού έχουμε τελειώσει με την δημιουργία του template, περνάμε τώρα στην δομή επανάληψης, όπου εκεί διαβάζουμε καινούργιες εικόνες και προσπαθούμε να σχεδιάσουμε το αντικείμενο που μοιάζει με το template μας στην εικόνα που διαβάζουμε.

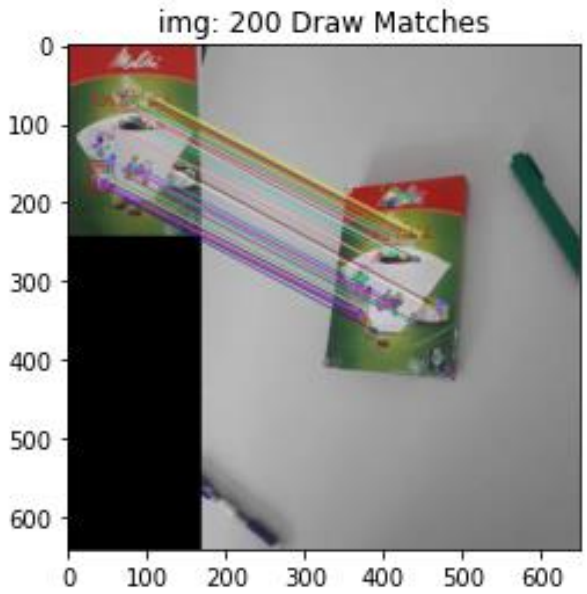
Αρχικά διαβάζουμε και εμφανίζουμε την εικόνα, στην συνέχεια βρίσκουμε και αντιστοιχούμε τα features (keypoints), από την οποία διαδικασία έχουμε επιλέξει να επιλεχθούν λιγότερες αντιστοιχίες, ορίζοντας ένα threshold = 40, το οποίο δηλαδή θα μας δώσει τις αντιστοιχίες keypoints, τα οποία έχουν απόσταση μικρότερη των 40 pixels. Με αυτόν τον τρόπο μας δίνει πιο λίγες αντιστοιχίες, οι οποίες όμως είναι καλύτερες. Δηλαδή προσπαθούμε να αφαιρέσουμε κάπως τον θόρυβο (λανθασμένες αντιστοιχίες).

Μόλις έχουμε τελειώσει με το στάδιο της αντιστοίχιση, προχωράμε στην απεικόνιση των αντιστοιχίσεων, όπου φαίνονται όλες οι αντιστοιχίσεις (Σωστές και Λάθος).

Παρακάτω εμφανίζονται οι αντιστοιχίες του template με την εικόνα 200.



Εικόνα 5



Εικόνα 6

```
Image: 200
Distance Range: 17.0 74.0
Keypoints template/image/mathces/filtered: 340 500 151 57
```

Εικόνα 7

Στην **Εικόνα 7** εμφανίζεται η Εικόνα που έχει χρησιμοποιηθεί, το εύρος της απόστασης μεταξύ των αντιστοιχισμένων keypoints που βρέθηκαν και τέλος εμφανίζεται ο αριθμός των keypoints (feautures) του template καθώς και της εικόνας και επιπλέον φαίνεται ο αριθμός των αντιστοιχίσεων (151) και ο αριθμός αυτών που πήραμε (57) λόγω του threshold που βάλαμε (να πάρουμε αντιστοιχίσεις που έχουν απόσταση μικρότερη των 40 pixels).

Έχοντας τώρα τα σημεία των αντιστοιχίσεων, ήρθε η στιγμή να βρούμε τους μετασχηματισμούς και να σχεδιάσουμε το αντικείμενο στην εικόνα μετασχηματίζοντας τα σημεία του template.

Μετασχηματισμός Affine

Αρχικά, δίνουμε τα σημεία σε μια συνάρτηση solve_affine. Αυτή η συνάρτηση έχει φτιαχτεί με τρόπο τέτοιον, ώστε να της δίνουμε σημεία keypoints του template και σημεία keypoints της εικόνας και εκείνη να επιστρέφει έναν μετασχηματισμό που τα συνδέει. Ο οποίος μετασχηματισμός θα έχει την παρακάτω μορφή.

$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}$$

Εικόνα 8

Η εύρεση αυτού του μετασχηματισμού, βασίζεται στον παρακάτω πίνακα:

$$\begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix}$$

Εικόνα 9

Στην περίπτωσή μας παίρνουμε σαν αποτέλεσμα τα παρακάτω:

Μετασχηματισμός affine που μετασχηματίζει τα σημεία του template σε σημεία της εικόνας.

```
affine without ransac:
 [[  0.91   -0.11 176.3 ]
 [  0.03    0.97 168.68]
 [  0.      0.     1.  ]]
```

Εικόνα 10

Εικόνα 11

Μετασχηματίζοντας τα σημεία του template με την χρήση του μετασχηματισμού affine που βρέθηκε προηγουμένως, παίρνουμε σαν αποτέλεσμα την **Εικόνα 9**, στην οποία εμφανίζεται η εικόνα 200 και σχεδιάζεται η θέση του template. Παρατηρούμε λοιπόν ότι έχει βρει το αντικείμενο που θέλαμε να βρούμε, αλλά οι διαστάσεις που έχουμε σχεδιάσει δεν περιγράφουν τόσο καλά τις διαστάσεις του αντικειμένου.

Αλγόριθμος RANSAC και εύρεση affine Μετασχηματισμού με την χρήση του αλγορίθμου RANSAC

Προχωράμε λοιπόν στην χρήση του αλγορίθμου RANSAC. Στην συγκεκριμένη περίπτωση για την εύρεση του καλύτερου μετασχηματισμού affine, υπάρχει μια δομή επανάληψη με αριθμό επαναλήψεων που εξαρτάτε από μια συγκεκριμένη συνάρτηση η οποία υπολογίζει τον απαιτούμενο αριθμό επαναλήψεων για το RANSAC. Η οποία παίρνει παραμέτρους:

- u: Πιθανότητα που μπορεί κάποιο σημείο να είναι εκτός γραμμής
- m: Ελάχιστος αριθμός ζευγαριών
- p: Επιθυμητή πιθανότητα για καλό δείγμα

και η οποία συμπεριλαμβάνει τον τύπο υπολογισμού των επαναλήψεων:

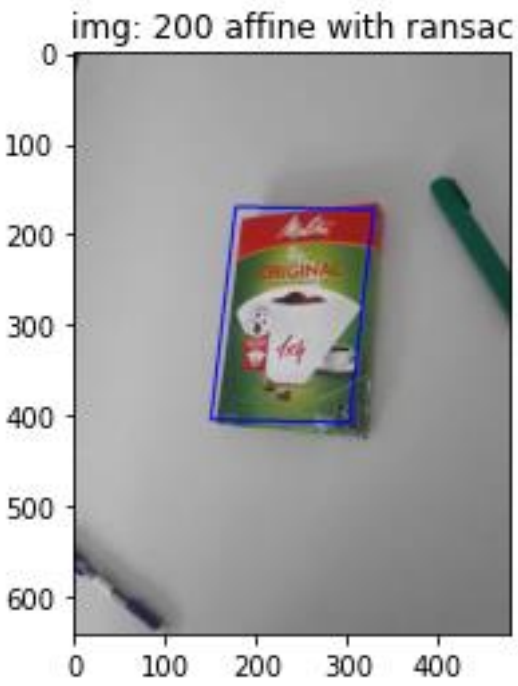
$$N = \frac{\log(1-p)}{\log(1-(1-v)^m)}$$

Στην περίπτωσή μας, οι παράμετροι p και u είναι σταθεροί για κάθε μετασχηματισμό, οι οποίοι είναι p = 0.99, u = 0.5 και η παράμετρος m εξαρτάτε από τον ελάχιστο αριθμό ζευγαριών. Στην περίπτωση του υπολογισμού του affine μετασχηματισμού η παράμετρος m είναι 3 και στην περίπτωση του υπολογισμού του Projective μετασχηματισμού η παράμετρος m είναι 4. Αυτό διότι χρειάζονται 3 ελάχιστα ζευγάρια για την επίλυση του πίνακα affine με 6 αγνώστους (3*2=6) και στην περίπτωση της εύρεσης του πίνακα μετασχηματισμού Projective, εκεί χρειάζονται 4 ζευγάρια (4*2=8). Οι άγνωστοι στην περίπτωση του υπολογισμού του μετασχηματισμού Projective είναι 9, αλλά μπορούμε να τον βρούμε και με 4 ζευγάρια (8 εξισώσεις).

Αφού έχουν οριστεί οι επαναλήψεις, περνάμε στο σημείο του τι γίνεται μέσα σε κάθε επανάληψη. Σε κάθε επανάληψη λαμβάνονται 3 σημεία αντιστοιχιών (template με εικόνα) και βρίσκεται (υπολογίζεται) ο μετασχηματισμός affine με την χρήση μόνο αυτών των $3^{ων}$ σημείων. Έπειτα εισερχόμαστε σε νέα συνάρτηση στην οποία μετασχηματίζονται τα σημεία (keypoints) του template με τον μετασχηματισμό που υπολογίστηκε προηγουμένως και υπολογίζεται η απόσταση των σημείων αυτών με τα σημεία (keypoints) που έχουν βρεθεί στην εικόνα.

Επίσης, υπάρχει ένα **threshold** το οποίο καθορίζει ποια απόσταση μεταξύ αυτών των σημείων είναι επιθυμητή και να καταμετρηθεί (καταμετράει όσες αποστάσεις είναι μικρότερες του threshold), το οποίο όταν είναι πολύ μεγάλο σημαίνει ότι παίρνει όλο και πιο μακρινά σημεία (λάθος σημεία) και καθώς μειώνεται παίρνει όλο και πιο σωστά σημείο. Απλώς μην είναι πολύ μικρό (1,2), διότι υπάρχει περίπτωση να μην δουλεύει σωστά (Στην άσκηση αυτή έχει χρησιμοποιηθεί threshold 3, διότι είχαμε καλύτερα αποτελέσματα με threshold 3). Αυτή η παραπάνω διαδικασία γίνεται N φορές και έχει σαν αποτέλεσμα να μας δώσει στο τέλος τον μετασχηματισμό affine που έχει τα περισσότερα inlier σημεία. Το οποίο σημαίνει ότι θα μας δώσει τον καλύτερο μετασχηματισμό. Με αυτόν τον τρόπο υλοποιήθηκε και η εύρεση του affine με την χρήση του αλγορίθμου RANSAC και προχωράμε να δούμε το αποτέλεσμα.

Εικόνα 12



Εικόνα 13

Στην περίπτωση αυτή επειδή έτυχε ο μετασχηματισμός χωρίς RANSAC να είναι σχεδόν ίδιο με τον μετασχηματισμό με RANSAC έχει σαν αποτέλεσμα 2 εικόνες που φαίνονται ίδιες, αλλά αν παρατηρήσετε καλύτερα οι άκρες του περιγράμματος έχουν αλλάξει ελάχιστα. Ας παρουσιάσουμε λοιπόν τους μετασχηματισμούς και τα αποτελέσματα των μετασχηματισμών, ώστε να φανούν καλύτερα οι διαφορές.

```
affine without ransac:
 [[  0.91   -0.11 176.3 ]
 [  0.03    0.97 168.68]
 [  0.      0.     1.  ]]
affine with ransac:
 [[  0.89   -0.11 177.69]
 [  0.02    0.96 170.93]
 [  0.      0.     1.  ]]
```

Εικόνα 14

```
Image Contour (affine without ransac):
 [[176 330 304 150]
 [168 173 409 404]]
Image Contour (affine with ransac):
 [[177 328 301 150]
 [170 173 407 404]]
```

Εικόνα 15

Όπως και να έχει, πάλι δεν μένουμε τόσο ικανοποιημένοι στον σχεδιασμό του αντικειμένου, διότι ο μετασχηματισμός affine έχει την ικανότητα να στρέφει, να μετατοπίζει και να αλλάζει την κλίμακα, δηλαδή μπορεί να μετατοπίζει τετράγωνα με παράλληλες πλευρές σε όποια θέση θέλουμε, με όποιον προσανατολισμό (σε 2D στην περίπτωση αυτή) θέλουμε και όσο μεγάλα θέλουμε να είναι (scale). Στην εικόνα όμως αυτή το αντικείμενό μας δεν έχει μορφή τετραγώνου με παράλληλες πλευρές, αλλά τραπεζοειδές μορφή θα έλεγε κάποιος.

Affine
6dof

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Εικόνα 16

Οπότε ο μετασχηματισμός affine δεν είναι κατάλληλος για τον σχεδιασμό του αντικειμένου. Πρέπει δηλαδή να βρεθεί ένας άλλος τύπος μετασχηματισμού, ο οποίος να μπορεί να κάνει αυτό που θέλουμε. Ένας άλλος τύπος μετασχηματισμού είναι η ομογραφία (Projective Transform).

Μετασχηματισμός Projective

Η ομογραφία ή ο μετασχηματισμός Projective έχειτην παρακάτω μορφή και την παρακάτω ικανότητα.

Projective
8dof

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Εικόνα 17

Οπότε δίνουμε τα σημεία σε μια συνάρτηση, η οποία είναι κατάλληλα κατασκευασμένη, ώστε να δέχεται σημεία του template και της εικόνας και να βρίσκει για τυχαία 4 σημεία του template και 4 σημεία της εικόνας τον μετασχηματισμό που τα συνδέει (Είναι 4 τα σημεία, διότι 4 είναι ο ελάχιστος αριθμός για να λυθεί το σύστημα και να βρεθεί ο μετασχηματισμός).

Η εύρεση του μετασχηματισμού Projective βασίζεται στον παρακάτω πίνακα:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

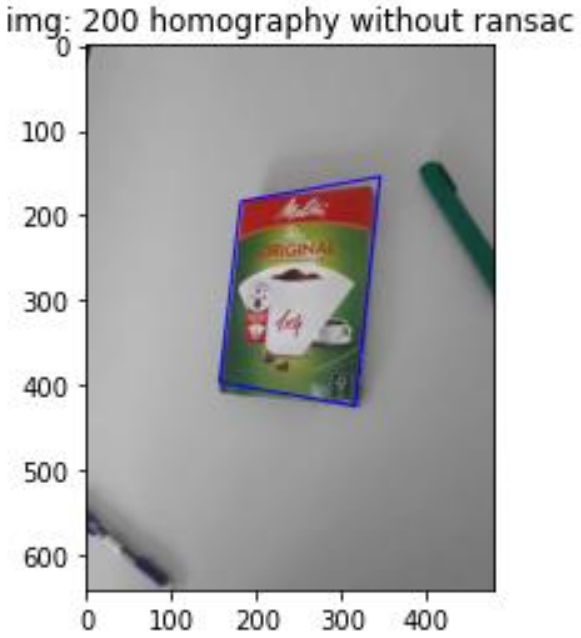
Εικόνα 18

Ο τυχαίος μετασχηματισμός που βρέθηκε είναι ο παρακάτω:

```
homography without ransac:
 [[-0.    0.   -0.7 ]
 [ 0.   -0.   -0.71]
 [ 0.    0.   -0.  ]]
```

Εικόνα 19

και η σχεδίαση του αντικειμένου με βάση αυτόν τον μετασχηματισμό είναι η παρακάτω:



Εικόνα 20

```
Image Contour (homography without ransac):
 [[181 344 315 156]
 [184 155 424 396]]
```

Εικόνα 21

Παρατηρείται λοιπόν ότι με τον μετασχηματισμό Projective (Ομογραφία) έχουμε σχεδιάσει το αντικείμενο πολύ καλύτερα από πριν και παρατηρείται ότι πλέον δεν έχουμε το πρόβλημα που είχαμε με τον μετασχηματισμό affine ο οποίος σχηματίζει τετράγωνα με παράλληλες γραμμές, εδώ μπορεί να πει κάποιος ότι έχουμε σχεδιάσει ένα τραπέζιο.

Στην περίπτωση αυτή έτυχε να πάρουμε ένα αρκετά καλό σχεδιασμό του αντικειμένου, αυτό διότι οι αντιστοιχίες που έγιναν μπορεί να πει κάποιος ότι είχαν αρκετά μεγάλο ποσοστό σωστής αντιστοίχισης. Εάν τώρα είχαμε λάθος αντιστοιχίσεις και βρίσκαμε τον μετασχηματισμό Projective, μπορεί να σχεδιάζαμε στην συνέχεια κάτι εντελώς άσχετο και ο λόγος επειδή τα σημεία δεν θα ήταν σωστά αντιστοιχισμένα.

Για να εξασφαλίσουμε ‘σωστό’ μετασχηματισμό, θα πρέπει να εφαρμόσουμε τον αλγόριθμο RANSAC και σ’ αυτήν την περίπτωση. Με αυτόν τον τρόπο, θα προσπαθήσουμε να πάρουμε εάν όχι τον καλύτερο, έναν αρκετά καλό (μπορεί και τον καλύτερο) μετασχηματισμό που προκύπτει από τα σημεία του template και της εικόνας.

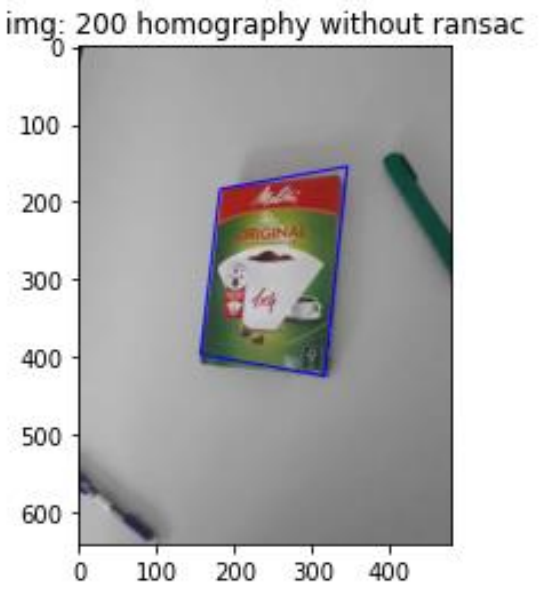
Εύρεση μετασχηματισμού* Projective *με την χρήση του αλγορίθμου RANSAC

Με την εφαρμογή του RANSAC στην εύρεση του μετασχηματισμού Projective, παίρνουμε τον παρακάτω μετασχηματισμό:

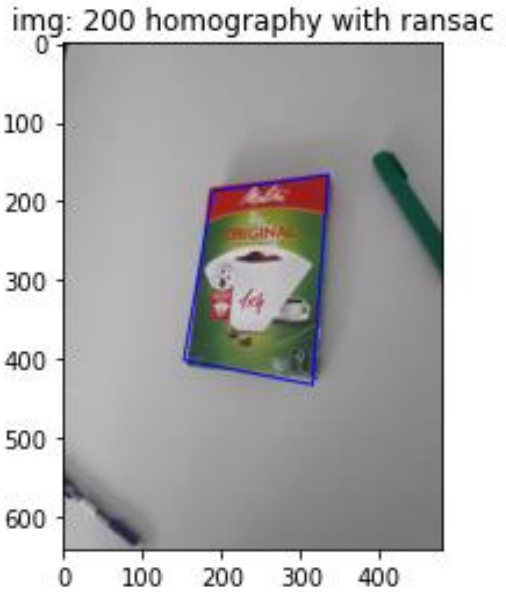
```
homography with ransac:
 [[-0.    0.   -0.71]
 [ 0.   -0.   -0.71]
 [ 0.    0.   -0.  ]]
```

Εικόνα 22

Ο μετασχηματισμός αυτός καθώς και ο προηγούμενος φαίνεται να έχει μηδενικά, αλλά κάτι τέτοιο δεν ισχύει. Υπάρχουν αριθμοί οι οποίοι όμως είναι πολύ μικροί και δεν εμφανίζονται ολόκληροι. Με την χρήση αυτού του μετασχηματισμού, ο σχεδιασμός που προκύπτει είναι ο παρακάτω, ο οποίος συγκρίνεται και με χωρίς RANSAC:



Εικόνα 23



Εικόνα 24

```
Image Contour (homography without ransac):
 [[181 344 315 156]
 [184 155 424 396]]
Image Contour (homography with ransac):
 [[188 334 314 152]
 [188 167 432 401]]
```

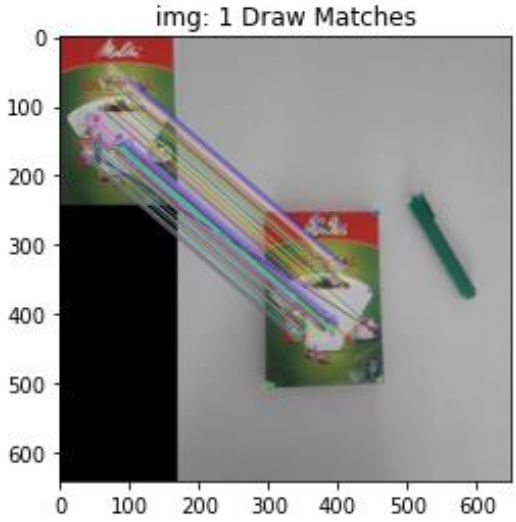
Εικόνα 25

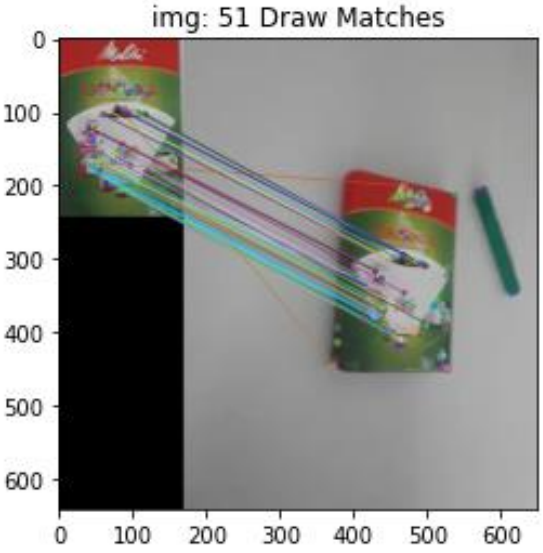
Παρατηρούμε ότι και οι 2 μετασχηματισμοί μπορούν να θεωρηθούν καλοί, διότι στην περίπτωση χωρίς τον RANSAC χρησιμοποιήθηκαν αρκετά καλά 4 σημεία για την εύρεση του μετασχηματισμού. Παρακάτω που θα υπάρξουν περισσότερα παραδείγματα, εκεί θα είναι εμφανές οι διαφορές μεταξύ του μετασχηματισμού affine χωρίς RANSAC και με RANSAC, καθώς επίσης και με μεταξύ του μετασχηματισμού Projective χωρίς και με RANSAC, όπου στην περίπτωση αυτή θα είναι τεράστια η αλλαγή εάν υπάρχουν αρκετές λάθος αντιστοιχίσεις (matching).

Προχωράμε λοιπόν σε μερικά επιπλέον παραδείγματα, στα οποία θα συμπεριληφθεί και βίντεο παρακολούθησης.

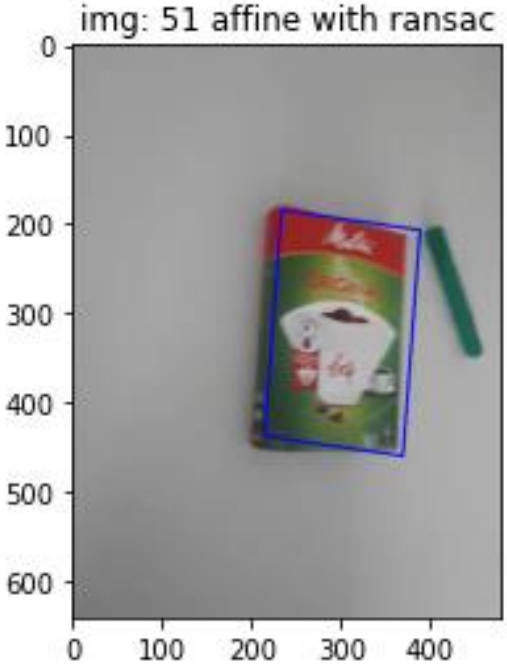
Παραδείγματα και Βίντεο

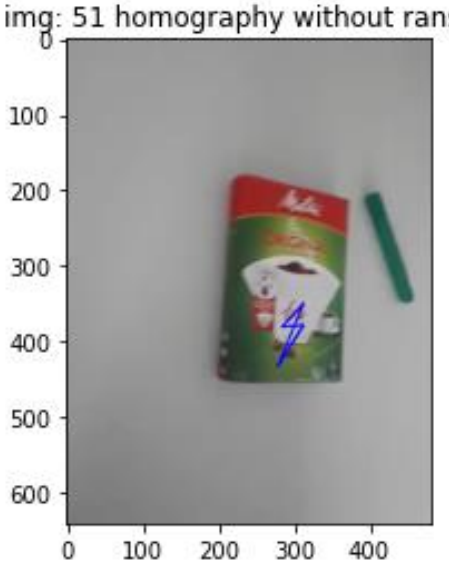
Παραδείγματα με εικόνες

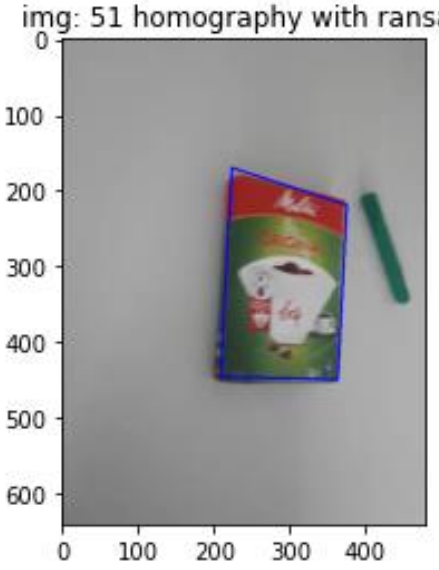


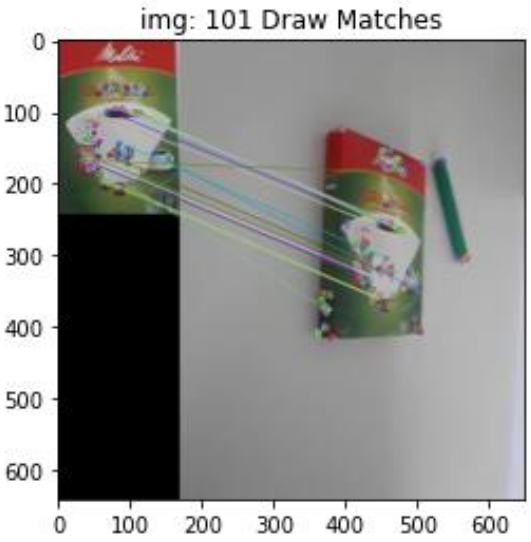

img: 51 Draw Matches

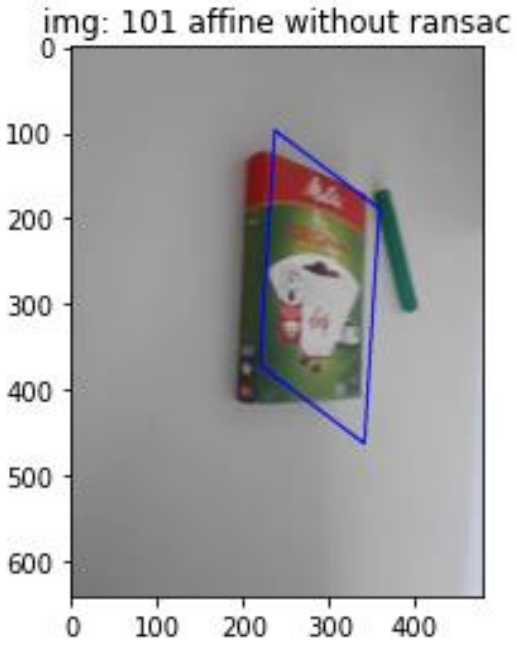

img: 51 affine without ransac

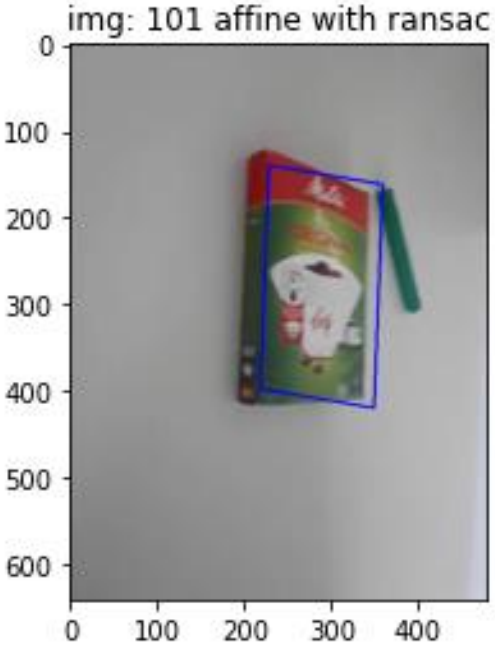

img: 51 affine with ransac

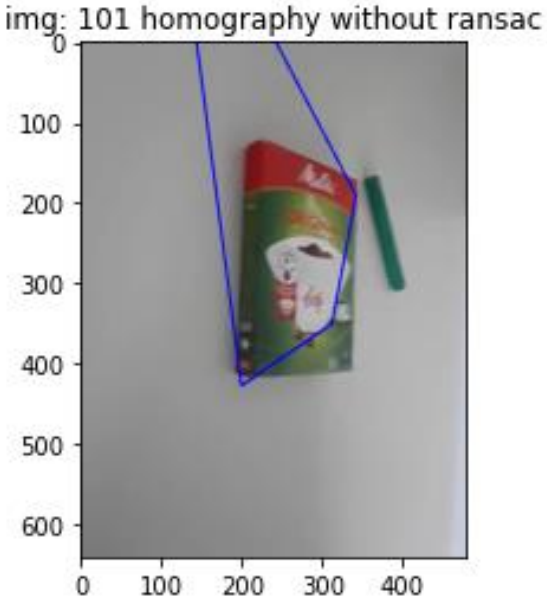

img: 51 homography without ransac


img: 51 homography with ransac

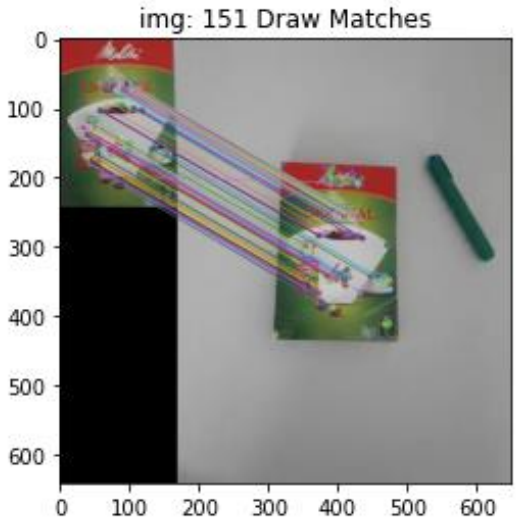

img: 101 Draw Matches
0
100
200
300
400
500
600
0
100
200
300
400
500
600


img: 101 affine without ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 101 affine with ransac
0
100
200
300
400
500
600
0
100
200
300
400

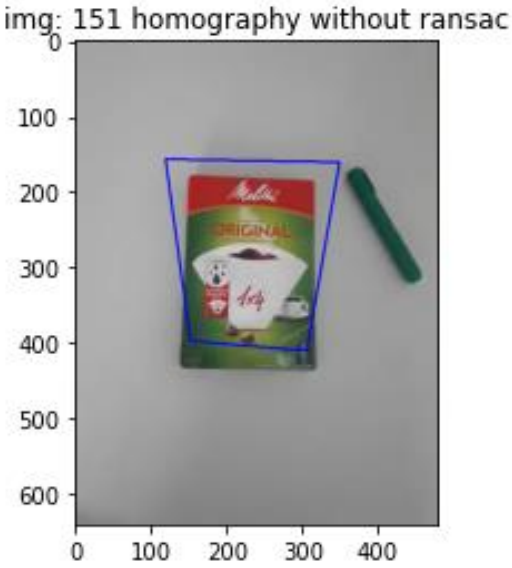

img: 101 homography without ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 101 homography with ransac
0
100
200
300
400
500
600
0
100
200
300
400

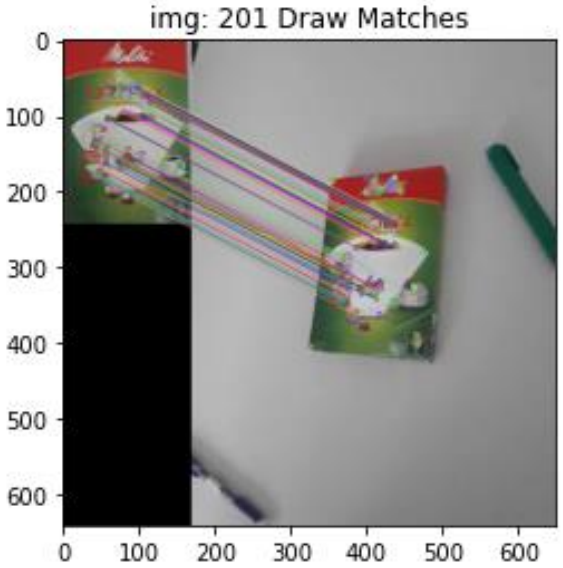

img: 151 Draw Matches


img: 151 affine without ransac

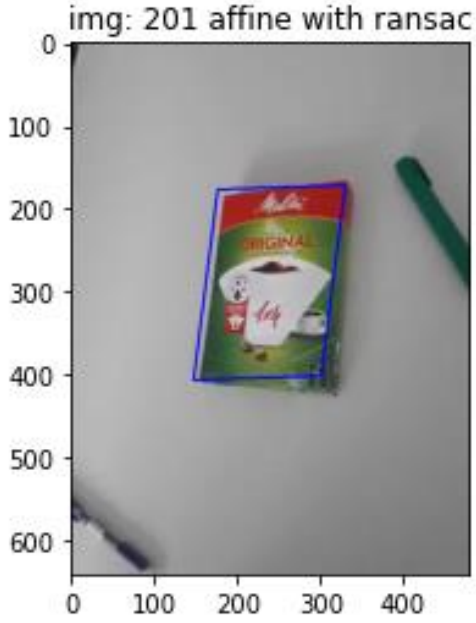

img: 151 affine with ransac


img: 151 homography without ransac


img: 151 homography with ransac

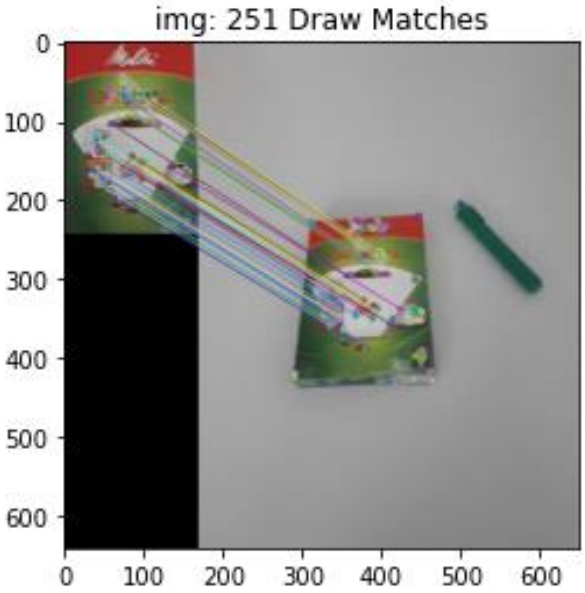

img: 201 Draw Matches


img: 201 affine without ransac


img: 201 affine with ransac


img: 201 homography without ransac


img: 201 homography with ransac

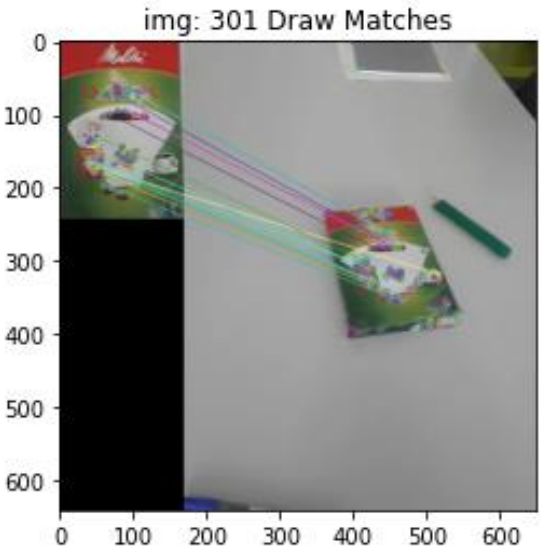

img: 251 Draw Matches

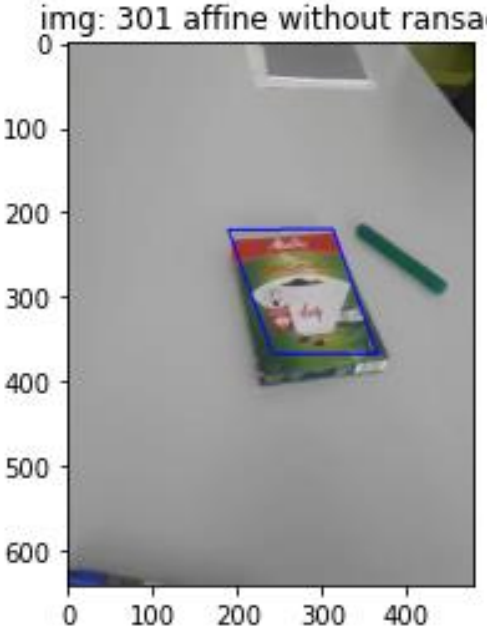

img: 251 affine without ransac

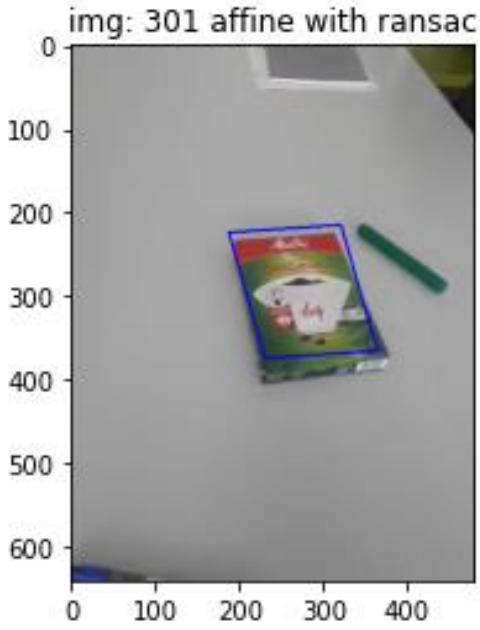

img: 251 affine with ransac

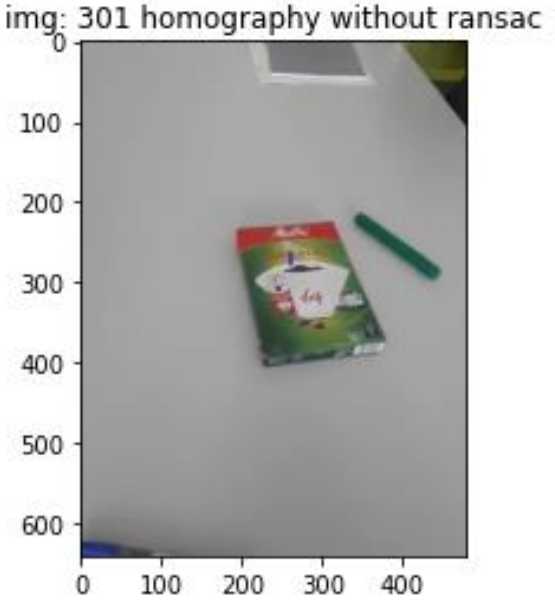

img: 251 homography without ransac


img: 251 homography with ransac

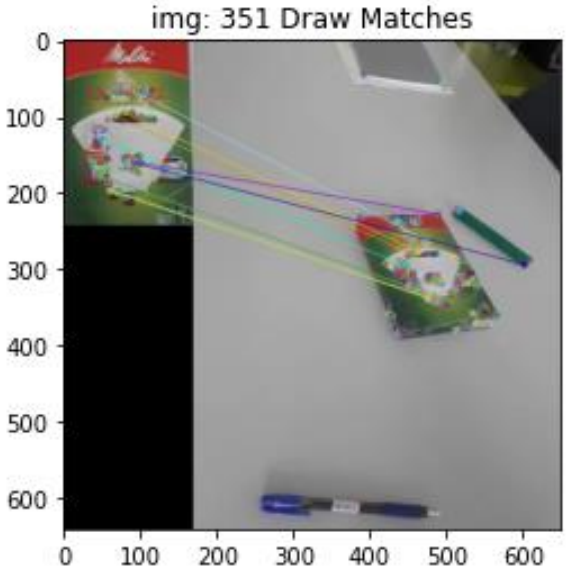

img: 301 Draw Matches
0
100
200
300
400
500
600
0
100
200
300
400
500
600

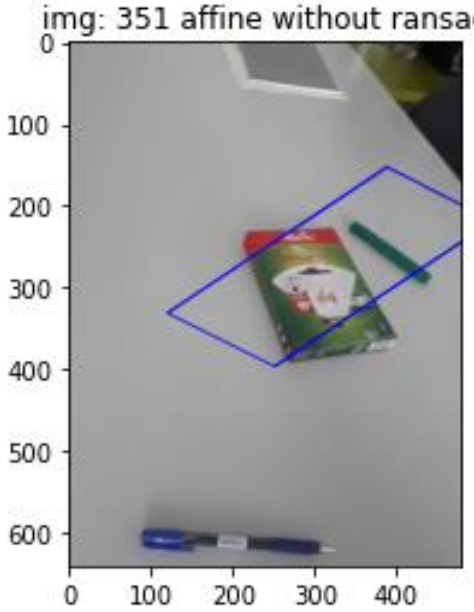

img: 301 affine without ransac
0
100
200
300
400
500
600
0
100
200
300
400

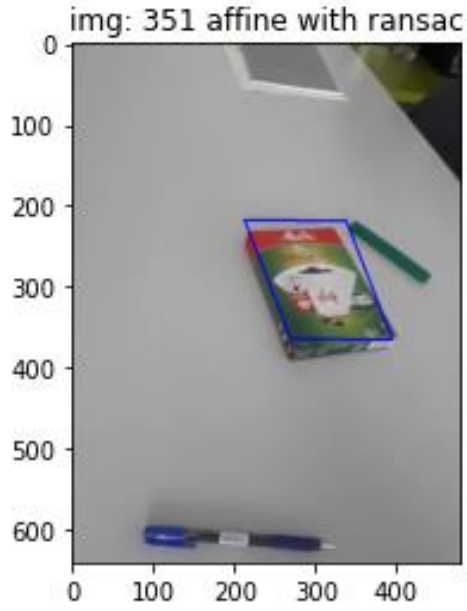

img: 301 affine with ransac
0
100
200
300
400
500
600
0
100
200
300
400

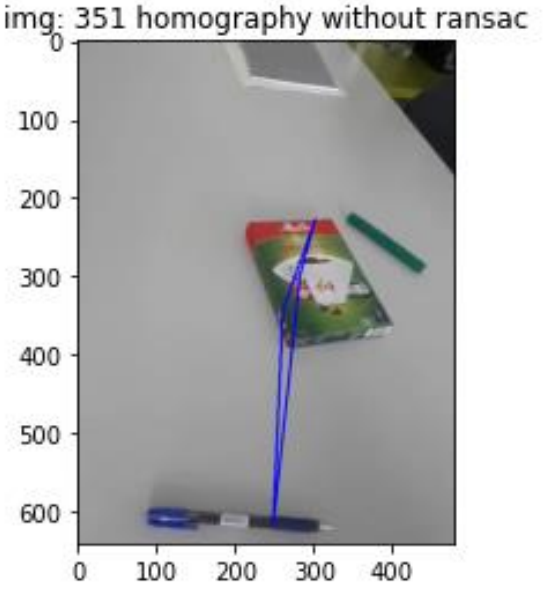

img: 301 homography without ransac
0
100
200
300
400
500
600
0
100
200
300
400

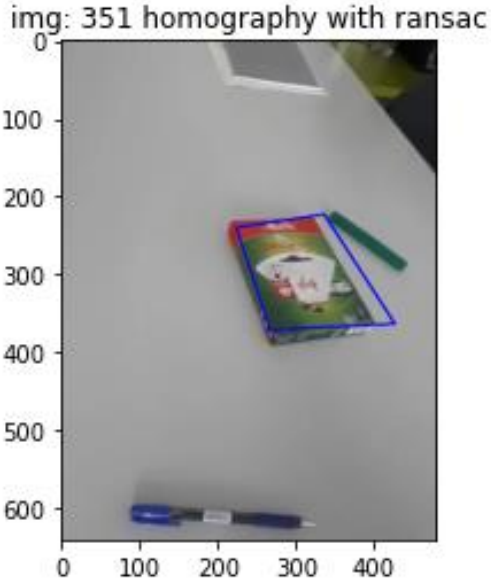

img: 301 homography with ransac
0
100
200
300
400
500
600
0
100
200
300
400

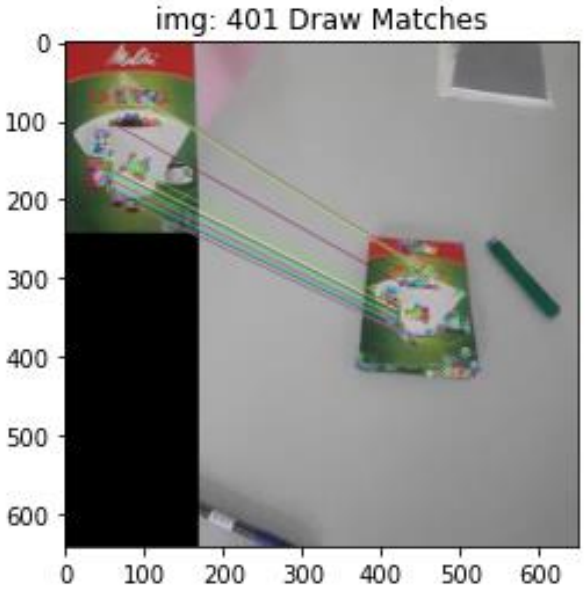

img: 351 Draw Matches
0
100
200
300
400
500
600
0
100
200
300
400
500
600

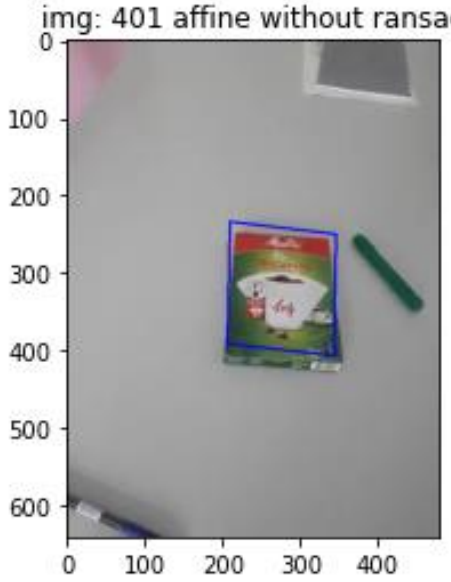

img: 351 affine without ransac
0
100
200
300
400
500
600
0
100
200
300
400

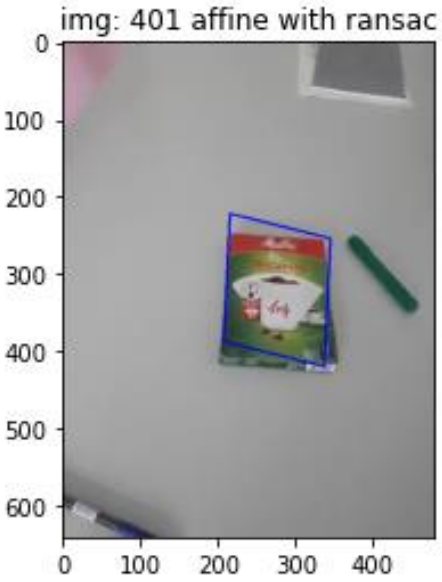

img: 351 affine with ransac
0
100
200
300
400
500
600
0
100
200
300
400

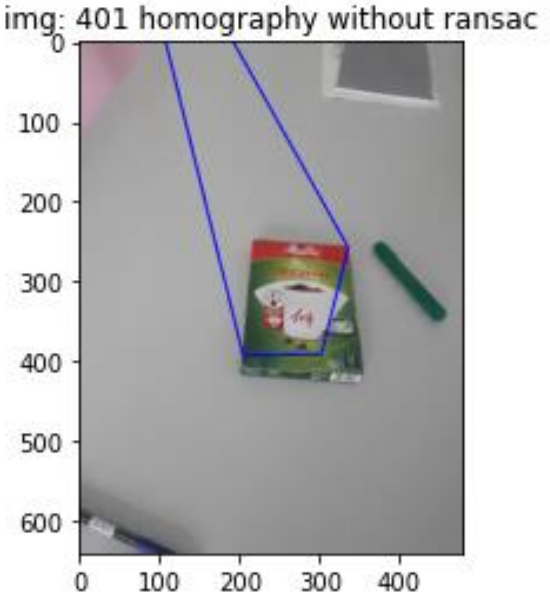

img: 351 homography without ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 351 homography with ransac
0
100
200
300
400
500
600
0
100
200
300
400

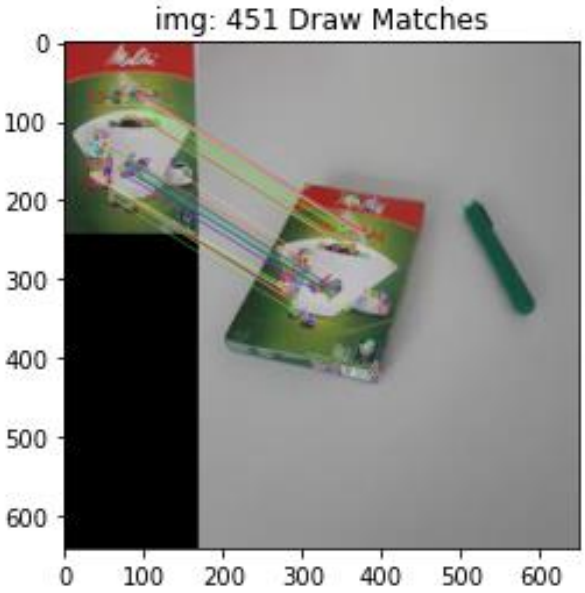

img: 401 Draw Matches
0
100
200
300
400
500
600
0 100 200 300 400 500 600


img: 401 affine without ransac
0
100
200
300
400
500
600
0 100 200 300 400

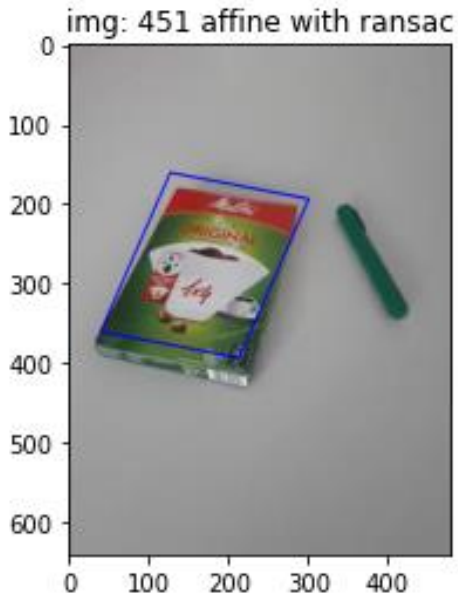

img: 401 affine with ransac
0
100
200
300
400
500
600
0 100 200 300 400

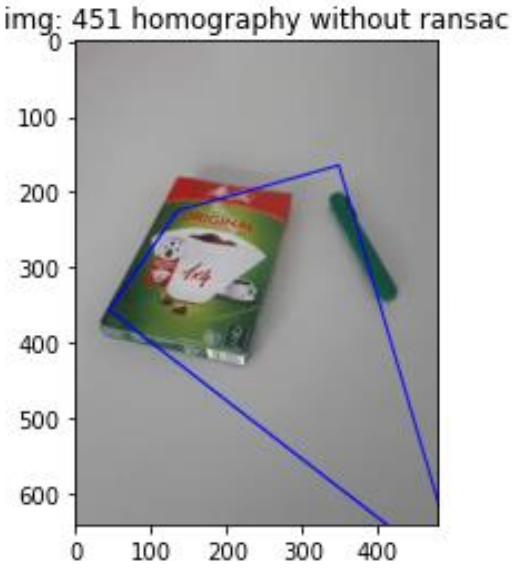

img: 401 homography without ransac
0
100
200
300
400
500
600
0 100 200 300 400

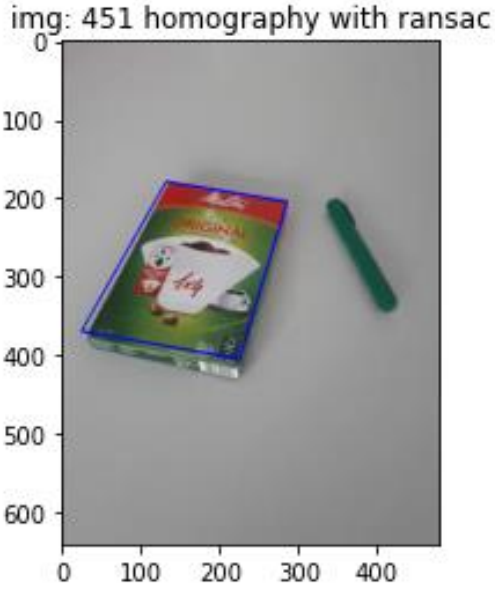

img: 401 homography with ransac
0
100
200
300
400
500
600
0 100 200 300 400


img: 451 Draw Matches
0
100
200
300
400
500
600
0
100
200
300
400
500
600


img: 451 affine without ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 451 affine with ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 451 homography without ransac
0
100
200
300
400
500
600
0
100
200
300
400


img: 451 homography with ransac
0
100
200
300
400
500
600
0
100
200
300
400

Αποτελέσματα μετασχηματισμών και σημείων περιγράμματος για την κάθε εικόνα που χρησιμοποιήθηκε προηγουμένως

```
Image: 1
Distance Range: 0.0 87.0
Keypoints template/image/mathces/filtered: 340 500 184 146
affine without ransac:
 [[  0.97   0.02 125.63]
 [  0.     1.   253.73]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  1.02   0.02 120.88]
 [  0.     0.99 255.3 ]
 [  0.     0.     1.  ]]
homography without ransac:
 [[ 0.   -0.    0.47]
 [-0.    0.    0.88]
 [-0.   -0.    0.  ]]
homography with ransac:
 [[-0.    0.   -0.44]
 [-0.   -0.   -0.9 ]
 [-0.    0.   -0.  ]]
```

```
Image Contour (affine without ransac):
 [[125 290 295 130]
 [253 254 497 496]]
Image Contour (affine with ransac):
 [[120 294 298 124]
 [255 255 497 496]]
Image Contour (homography without ransac):
 [[166 297 358 141]
 [316 236 576 492]]
Image Contour (homography with ransac):
 [[123 291 294 120]
 [251 259 497 502]]
```

```
Image: 51
Distance Range: 17.0 72.0
Keypoints template/image/mathces/filtered: 340 500 134 55
affine without ransac:
 [[  0.88   0.04 218.35]
 [  0.1    0.82 219.8 ]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.91  -0.08 233.23]
 [  0.13   1.04 185.12]
 [  0.     0.     1.  ]]
homography without ransac:
 [[ 0.   -0.01  0.6 ]
 [ 0.   -0.01  0.8 ]
 [ 0.   -0.    0.  ]]
homography with ransac:
 [[-0.    0.   -0.8 ]
 [-0.   -0.   -0.61]
 [-0.    0.   -0.  ]]
```

```
Image Contour (affine without ransac):
 [[218 367 377 228]
 [219 237 435 418]]
Image Contour (affine with ransac):
 [[233 387 368 214]
 [185 207 459 437]]
Image Contour (homography without ransac):
 [[284 312 278 312]
 [380 349 432 378]]
Image Contour (homography with ransac):
 [[223 374 362 208]
 [170 219 449 445]]
```

```
Image: 101
Distance Range: 16.0 77.0
Keypoints template/image/mathces/filtered: 340 500 108 24
affine without ransac:
 [[  0.71  -0.07 236.25]
 [  0.54   1.12  98.66]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.77  -0.04 228.27]
 [  0.13   1.06 140.89]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.04 -0.01 -0.41]
 [-0.03 -0.02  0.91]
 [-0.   -0.   -0.  ]]
homography with ransac:
 [[-0.    0.   -0.87]
 [-0.   -0.   -0.5 ]
 [-0.    0.   -0.  ]]
```

```
Image Contour (affine without ransac):
 [[236 357 340 219]
 [ 98 189 462 371]]
Image Contour (affine with ransac):
 [[228 358 348 218]
 [140 162 419 397]]
Image Contour (homography without ransac):
 [[ 111  341  311  199]
 [-250  191  350  427]]
Image Contour (homography with ransac):
 [[230 338 336 208]
 [132 188 403 420]]
```

```
Image: 151
Distance Range: 11.0 77.0
Keypoints template/image/mathces/filtered: 340 500 162 79
affine without ransac:
 [[  1.03   -0.04 147.56]
 [  0.04   0.92 185.8 ]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  1.04   -0.03 145.11]
 [  0.03   1.   178.35]
 [  0.     0.     1.  ]]
homography without ransac:
 [[ 0.01  0.    0.61]
 [-0.    0.01  0.8 ]
 [-0.    0.    0.01]]
homography with ransac:
 [[0.01 0.   0.64]
 [0.   0.01 0.76]
 [0.   0.   0.  ]]
Image Contour (affine without ransac):
 [[147 321 312 138]
 [185 191 415 409]]
Image Contour (affine with ransac):
 [[145 321 313 137]
 [178 183 426 421]]
Image Contour (homography without ransac):
 [[119 348 305 152]
 [157 161 410 397]]
Image Contour (homography with ransac):
 [[139 327 304 144]
 [165 180 421 412]]
```

```
Image: 201
Distance Range: 16.0 83.0
Keypoints template/image/mathces/filtered: 340 500 158 57
affine without ransac:
 [[  0.9   -0.12 176.11]
 [  0.02   0.94 172.62]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.9   -0.12 176.42]
 [ -0.03   0.94 177.66]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.    -0.    0.51]
 [-0.01 -0.    0.86]
 [-0.    -0.    0.  ]]
homography with ransac:
 [[ 0.    -0.    0.71]
 [-0.    0.    0.71]
 [-0.    -0.    0.  ]]
Image Contour (affine without ransac):
 [[176 329 300 146]
 [172 175 404 401]]
Image Contour (affine with ransac):
 [[176 329 300 147]
 [177 172 401 406]]
Image Contour (homography without ransac):
 [[200 180 199 194]
 [336 364 338 345]]
Image Contour (homography with ransac):
 [[183 336 308 149]
 [184 166 423 401]]
```

```
Image: 251
Distance Range: 15.0 83.0
Keypoints template/image/mathces/filtered: 340 500 128 47
affine without ransac:
 [[  0.96   -0.04 132.86]
 [ -0.03   0.8  214.59]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.94   -0.04 134.03]
 [ -0.02   0.82 211.1 ]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.    0.   -0.55]
 [ 0.    0.   -0.83]
 [ 0.    0.   -0.  ]]
homography with ransac:
 [[ 0.    -0.    0.53]
 [-0.    0.    0.85]
 [-0.    -0.    0.  ]]
Image Contour (affine without ransac):
 [[132 295 285 122]
 [214 209 404 409]]
Image Contour (affine with ransac):
 [[134 293 283 123]
 [211 207 406 410]]
Image Contour (homography without ransac):
 [[160 272 314  99]
 [243 225 439 446]]
Image Contour (homography with ransac):
 [[142 282 297 113]
 [227 220 416 420]]
```

```
Image: 301
Distance Range: 14.0 80.0
Keypoints template/image/mathces/filtered: 340 500 111 16
affine without ransac:
 [[  0.72   0.21 189.57]
 [ -0.     0.6  221.31]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.78   0.18 187.7 ]
 [ -0.06   0.62 224.79]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.   -0.    0.7 ]
 [-0.01 -0.    0.71]
 [-0.   -0.    0.  ]]
homography with ransac:
 [[0.01 0.   0.64]
 [0.   0.   0.77]
 [0.   0.   0.  ]]
Image Contour (affine without ransac):
 [[189 312 364 241]
 [221 220 367 367]]
Image Contour (affine with ransac):
 [[187 320 363 230]
 [224 215 365 374]]
Image Contour (homography without ransac):
 [[260 260 260 260]
 [264 269 263 279]]
Image Contour (homography with ransac):
 [[175 315 346 224]
 [211 223 353 387]]
```

```
Image: 351
Distance Range: 28.0 81.0
Keypoints template/image/mathces/filtered: 340 500 103 10
affine without ransac:
 [[  1.57   0.53 120.69]
 [ -1.04   0.27 331.06]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.74   0.23 212.43]
 [  0.     0.6  218.92]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.   -0.    0.69]
 [-0.   -0.    0.73]
 [-0.   -0.    0.  ]]
homography with ransac:
 [[ 0.   -0.    0.68]
 [-0.    0.    0.73]
 [-0.   -0.    0.  ]]
Image Contour (affine without ransac):
 [[120 387 516 250]
 [331 153 219 396]]
Image Contour (affine with ransac):
 [[212 337 394 269]
 [218 219 365 364]]
Image Contour (homography without ransac):
 [[282 301 260 248]
 [298 228 344 614]]
Image Contour (homography with ransac):
 [[223 335 425 266]
 [240 223 363 371]]
```

```
Image: 401
Distance Range: 21.0 80.0
Keypoints template/image/mathces/filtered: 340 500 109 17
affine without ransac:
 [[  0.8   -0.01 209.37]
 [  0.09   0.65 234.87]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  0.76  -0.03 215.62]
 [  0.18   0.68 222.64]
 [  0.     0.     1.  ]]
homography without ransac:
 [[-0.04 -0.01 -0.15]
 [-0.03 -0.03  0.99]
 [-0.   -0.   -0.  ]]
homography with ransac:
 [[-0.    0.   -0.65]
 [-0.   -0.   -0.76]
 [-0.    0.   -0.  ]]
Image Contour (affine without ransac):
 [[209 345 342 206]
 [234 250 406 391]]
Image Contour (affine with ransac):
 [[215 344 336 207]
 [222 254 419 387]]
Image Contour (homography without ransac):
 [[  41  335  301  204]
 [-270  258  391  392]]
Image Contour (homography with ransac):
 [[217 333 356 192]
 [253 255 405 408]]
```

```
Image: 451
Distance Range: 21.0 82.0
Keypoints template/image/mathces/filtered: 340 500 145 33
affine without ransac:
 [[  1.03  -0.34 124.68]
 [  0.19   0.81 163.55]
 [  0.     0.     1.  ]]
affine with ransac:
 [[  1.02  -0.36 127.56]
 [  0.2    0.82 161.23]
 [  0.     0.     1.  ]]
homography without ransac:
 [[ 0.   -0.    0.52]
 [-0.    0.    0.86]
 [-0.   -0.    0.  ]]
homography with ransac:
 [[-0.    0.   -0.59]
 [-0.   -0.   -0.8 ]
 [ 0.    0.   -0.  ]]
```

```
Image Contour (affine without ransac):
 [[124 300 217  41]
 [163 196 393 360]]
Image Contour (affine with ransac):
 [[127 300 213  40]
 [161 194 393 360]]
Image Contour (homography without ransac):
 [[135 347 509  45]
 [225 165 716 356]]
Image Contour (homography with ransac):
 [[132 285 224  25]
 [179 205 404 369]]
```

Βίντεο

Το βίντεο υπάρχει στον φάκελο Βίντεο (exercise_02_video.mp4), στο οποίο φαίνονται οι αντιστοιχίσεις καθώς και ο σχεδιασμός του αντικειμένου (Melita) με την χρήση των 2 μετασχηματισμών (Affine και Projective) με και χωρίς την χρήση του αλγορίθμου RANSAC. Επίσης στον φάκελο αυτόν, υπάρχουν και τα κομμάτια αυτού του βίντεο ξεχωριστά.

Τα αποτελέσματα του βίντεο καθώς και τα αποτελέσματα που αναφέρθηκαν προηγουμένως κατά την αναφορά – επεξήγηση του τρόπου επίλυσης της άσκησης 2, βασίζονται στις ίδιες παραμέτρους, δηλαδή δεν είχαν πειραχτεί παράμετροι που συσχετίζονται με το matching ή τον αλγόριθμο RANSAC και γενικότερα δεν είχε πειραχτεί τίποτα απολύτως που θα μπορούσε να αλλάξει τα αποτελέσματα. Το μόνο που άλλαζε ήταν τα αποτελέσματα που θέλαμε να εμφανίσουμε (Σχεδίαση μόνο του αντικειμένου με affine χωρίς RANSAC, ή σχεδίαση μόνο του αντικειμένου με RANSAC κ.λπ.).