

Όραση Υπολογιστών
**Εργασία 2η: Δημιουργία πανοράματος με
αλγορίθμους SIFT, SURF, ORB**

Γραμμένος-Γεώργιος Πολυμερίδης
Α.Μ.: 58105

30 Νοεμβρίου 2025

Περιεχόμενα

1 Πρόβλημα	5
1.1 Περιγραφή Προβλήματος	5
2 Εισαγωγή	6
3 Πρόβλημα και Σκοπός	6
4 Προκλήσεις στη Δημιουργία Πανοράματος	6
4.1 Γεωμετρικές Προκλήσεις	6
4.2 Ραδιομετρικές Προκλήσεις	6
5 Ζητούμενα της Εργασίας	7
5.1 Ζήτημα 1: Custom Implementation του Cross-Checking	7
5.2 Ζήτημα 2: Δημιουργία Πανοραμάτων με Τρεις Ανιχνευτές	7
5.3 Ζήτημα 3: Δημιουργία Πανοραμάτων με Τρεις Μεθόδους Προβολής	7
5.4 Ζήτημα 4: Σύγκριση Αποτελεσμάτων	8
5.5 Ζήτημα 5: Δοκιμές σε Δική σας Λήψη	8
6 Δομή της Αναφοράς	8
7 Ανιχνευτές Χαρακτηριστικών και Περιγραφείς	9
7.1 SIFT - Scale-Invariant Feature Transform	9
7.1.1 Αρχές του SIFT	9
7.1.2 Εφαρμογή στον Κώδικα	9
7.2 SURF - Speeded-Up Robust Features	9
7.2.1 Αρχές του SURF	10
7.2.2 Εφαρμογή στον Κώδικα	10
7.3 ORB - Oriented FAST and Rotated BRIEF	10
7.3.1 Αρχές του ORB	10
7.3.2 Πλεονεκτήματα και Μειονεκτήματα	10
7.3.3 Εφαρμογή στον Κώδικα	11
7.4 Σύγκριση των Τριών Αλγορίθμων	11
8 Διαδικασία Εξαγωγής Χαρακτηριστικών	11
9 Εξευρετικός Ταιριασμα Χαρακτηριστικών με BFMatcher	13
9.1 Brute-Force Matching	13
9.1.1 Αρχές του Αλγορίθμου	13
9.1.2 Μετρικές Απόστασης	13
9.2 Cross-Checking για Φιλτράρισμα Matches	13
9.2.1 Αρχή του Cross-Checking	13
9.2.2 Υλοποίηση του Cross-Checking	14
9.2.3 Διαδικασία Ταιριάσματος	14
9.3 Αποτελέσματα Ταιριάσματος	14
9.4 Σύγκριση με Lowe's Ratio Test	15

10 Μέθοδοι Προβολής για Stitching	16
10.1 Homography - Ο Προοπτικός Μετασχηματισμός	16
10.2 RANSAC - Random Sample Consensus	16
10.2.1 Αρχή του RANSAC	16
10.3 Planar Stitching - Προοπτικό Stitching	17
10.3.1 Διαδικασία	17
10.4 Cylindrical Stitching - Κυλινδρική Προβολή	17
10.4.1 Μαθηματικά της Κυλινδρικής Προβολής	17
10.4.2 Υλοποίηση στον Κώδικα	18
10.4.3 Πλεονεκτήματα Κυλινδρικής Προβολής	18
10.5 Hybrid Stitching - Υβριδική Προβολή	19
10.5.1 Στρατηγική	19
10.6 Τελικό Stitching - Perspective Transform	19
10.7 Σύγκριση των Τριών Μεθόδων Προβολής	20
11 Αποτελέσματα και Οπτικοποίηση Πανοραμάτων	21
11.1 Περιγραφή Datasets	21
11.2 Αποτελέσματα Panorama Stitching - Σύγκριση με ICE	22
11.2.1 NISwGP-02 Uffizi Dataset	22
11.2.2 OpenPano Medium (Outdoor Landscape)	23
11.2.3 UAV Orthophotomap (Aerial Drone Photography)	23
11.2.4 Ποιοτική Σύγκριση: Ours vs ICE	25
11.3 Προβλήματα και Περιορισμοί του warpPerspective	27
11.3.1 Μεγάλες Γωνίες και Memory Corruption	27
11.3.2 Παράδειγμα: NISwGP-02 Uffizi με Μεγάλες Γωνίες	27
11.3.3 Γιατί Cylindrical Projection Λύνει το Πρόβλημα	27
11.4 Σύγκριση Αλγορίθμων Feature Detection (SIFT vs SURF vs ORB)	29
11.4.1 SURF - Ταχύτητα με Αποδεκτή Ακρίβεια	29
11.4.2 ORB - Extreme Speed, Acceptable Quality	29
11.4.3 Comprehensive Algorithm Comparison Table	30
12 Ανάλυση Κώδικα - Υλοποίηση των Αλγορίθμων	31
12.1 Feature Matching με Lowe's Ratio Test	31
12.2 Cylindrical Warping - Προβολή σε Κύλινδρο	31
12.3 Planar Stitching - Ομογραφία	31
12.4 Hybrid Approach - Cylindrical + Planar	31
12.5 Crop Image - Αφαίρεση Μαύρων Περιθωρίων	32
13 Ανάλυση Αποτελεσμάτων και Σύγκριση	33
13.1 Ποσοτική Σύγκριση Αλγορίθμων	33
13.2 Ποσοτική Σύγκριση Προβολών	33
13.3 Επιλογές για Διαφορετικές Εφαρμογές	33
13.4 Προκλήσεις και Περιορισμοί	33
13.4.1 Περιοδική Υφή (UAV Dataset)	33
13.4.2 Φωτισμός και Σκιές	33
13.4.3 Μεγάλες Κινήσεις	33

14 Βέλτιστες Πρακτικές (Best Practices)	34
14.1 Προ-επεξεργασία Εικόνων	34
14.2 Feature Matching	34
14.3 Homography Refinement	34
14.4 Post-processing	34
15 Ανακεφαλαίωση Αποτελεσμάτων	35
15.1 Κυρίαρχα Συμπεράσματα	35
15.2 Συστάσεις για Μελλοντικές Εργασίες	35
16 Συμπεράσματα	36
16.1 Σύνοψη Εργασίας	36
16.2 Κύρια Ευρήματα	36
16.2.1 Ταίριασμα Χαρακτηριστικών	36
16.2.2 Μέθοδοι Προβολής	36
16.2.3 Σύγκριση με Commercial Software	36
16.3 Ανάλυση Αποδοσης	37
16.3.1 Ταχύτητα Εκτέλεσης	37
16.3.2 Ρυθμός Επιτυχίας	37
16.4 Περιορισμοί της Υλοποίησης	37
16.4.1 Θέματα που δεν Αντιμετωπίστηκαν	37
16.5 Μελλοντικές Κατευθύνσεις	37
16.5.1 Τεχνικές Βελτιώσεις	37
16.5.2 Επέκταση σε 3D	37
16.5.3 Εφαρμογές Real-World	38
16.6 Συστάσεις	38
16.6.1 Για Ακρίβεια	38
16.6.2 Για Ταχύτητα	38
16.6.3 Για Γενικές Εφαρμογές	38
16.6.4 Αποφεύγοντας	38
17 Τελικές Σκέψεις	38

1 Πρόβλημα

1.1 Περιγραφή Προβλήματος

Το πρόβλημα που αντιμετωπίζουμε η δημιουργία πανοράματος από πολλές εικόνες. Θα κάνω κάποιες παραδοχές για την πρώτη προσέγγιση του προβλήματος

- Οι εικόνες εφόσον χρησιμοποιήθηκε ο ίδιος φακός, θα έχουν συνέπεια διαστάσεων
- Δεν γνωρίζω τις διαστάσεις του φακού ή μπορεί να έχω ή να μην έχω πληροφορίες
- Η λήψεις είναι ικανοποιητικές

Παρατηρούμε ότι όσο αυξάνεται το kernel size τόσο πιο εύκολα συγχωνεύονται μεγάλα κενά διαστήματα, με αποτέλεσμα να χάνουμε την ακρίβεια στην αναγνώριση και αρίθμιση των υποπεριοχών.

2 Εισαγωγή

3 Πρόβλημα και Σκοπός

Η δημιουργία πανοράματος από πολλαπλές εικόνες είναι ένα θεμελιώδες πρόβλημα στη Μηχανική Όρασης (Computer Vision). Σε αυτή την εργασία θα αναπτύξουμε και θα συγκρίνουμε τρεις δημοφιλείς αλγορίθμους για ανίχνευση χαρακτηριστικών γνωρισμάτων:

1. SIFT (Scale-Invariant Feature Transform)
2. SURF (Speeded-Up Robust Features)
3. ORB (Oriented FAST and Rotated BRIEF)

4 Προκλήσεις στη Δημιουργία Πανοράματος

Η ενσωμάτωση πολλαπλών εικόνων σε ένα ενιαίο πανόραμα παρουσιάζει πολλές τεχνικές προκλήσεις:

4.1 Γεωμετρικές Προκλήσεις

Parallax Όταν λαμβάνουμε εικόνες από διαφορετικές θέσεις, τα αντικείμενα σε διαφορετικές αποστάσεις από την κάμερα εμφανίζονται να κινούνται σχετικά το ένα ως το άλλο.

Drift Σφάλματα συσσωρεύονται καθώς ράβουμε περισσότερες εικόνες σε σειρά, ιδίως στην κάθετη κατεύθυνση.

Distortion Οι αποχρώσεις και παραμορφώσεις του φακού πρέπει να διορθωθούν πριν το stitching.

4.2 Ραδιομετρικές Προκλήσεις

Illumination Changes Οι αλλαγές φωτεινότητας μεταξύ εικόνων προκαλούν ορατές ραφές.

Exposure Differences Διαφορές στο shutter speed ή ISO επηρεάζουν τη λαμπρότητα κάθε εικόνας.

Vignetting Σκοτείνιασμα στις άκρες τις εικόνας που πρέπει να ληφθεί υπόψη.

5 Ζητούμενα της Εργασίας

5.1 Ζήτημα 1: Custom Implementation του Cross-Checking

Υλοποίηση του αλγορίθμου cross-checking για το φιλτράρισμα ταιριασμάτων χαρακτηριστικών χωρίς τη χρήση της ενσωματωμένης συνάρτησης BFMatcher(crossCheck=True) της OpenCV.

Στόχος:

- Κατανόηση του Mutual Nearest Neighbors (MNN) αλγορίθμου
- Σύγκριση με Lowe's Ratio Test
- Αξιολόγηση επίδρασης στη ποιότητα stitching

5.2 Ζήτημα 2: Δημιουργία Πανοραμάτων με Τρεις Ανιχνευτές

Υλοποίηση δημιουργίας πανοραμάτων χρησιμοποιώντας:

- SIFT (Scale-Invariant Feature Transform)
- SURF (Speeded Up Robust Features)
- ORB (Oriented FAST and Rotated BRIEF)

Για κάθε ανιχνευτή, δημιουργία πανοραμάτων σε τρία datasets:

- NISwGP-02 Uffizi
- OpenPano Medium
- UAV Orthophotomap

5.3 Ζήτημα 3: Δημιουργία Πανοραμάτων με Τρεις Μεθόδους Προβολής

Υλοποίηση τριών διαφορετικών μεθόδων προβολής:

- Planar (Projective) Stitching
- Cylindrical Stitching
- Hybrid Stitching

Στόχος: Σύγκριση ποιότητας και απόδοσης των τριών μεθόδων

5.4 Ζήτημα 4: Σύγκριση Αποτελεσμάτων

Σύγκριση όλων των συνδυασμών: $3 \text{ Ανιχνευτές} \times 3 \text{ Μέθοδοι Προβολής} \times 3 \text{ Datasets} = 27$ πανοράματα

Κριτήρια σύγκρισης:

- Ποιότητα πανοραμάτων
- Artifacts (ghosting, misalignment)
- Robustness
- Computational cost
- Σύγκριση με εμπορικό λογισμικό (ICE)

5.5 Ζήτημα 5: Δοκιμές σε Δική σας Λήψη

Λήψη 4 εικόνων με κάμερα και δοκιμή του αλγορίθμου:

- Πρώτη δοκιμή: Ιδανικές συνθήκες (μικρή περιστροφή, μεγάλη επικάλυψη)
- Δεύτερη δοκιμή: Διαφορετικός προσανατολισμός κάμερας
- Τρίτη δοκιμή: Μικρή επικάλυψη (15-25%)

Ανάλυση: Παρατηρήσεις και Συμπεράσματα σχετικά με αξιοπιστία

6 Δομή της Αναφοράς

Η αναφορά αυτή διαρθρώνεται ως εξής:

- Κεφάλαιο 2: Ανιχνευτές και Περιγραφείς Χαρακτηριστικών (SIFT, SURF, ORB)
- Κεφάλαιο 3: Custom Implementation του Cross-Checking
- Κεφάλαιο 4: Μέθοδοι Προβολής (Planar, Cylindrical, Hybrid)
- Κεφάλαιο 5: Αποτελέσματα Πανοραμάτων
- Κεφάλαιο 6: Ανάλυση και Συμπεράσματα

7 Ανίχνευσης Χαρακτηριστικών και Περιγραφές

Η ανίχνευση και αντιστοίχιση χαρακτηριστικών είναι ο θεμελιώδης σταθμός για δημιουργία πανοραμάτων. Σε αυτό το κεφάλαιο περιγράφουμε τρεις δημοφιλείς αλγορίθμους που χρησιμοποιούμε σε αυτή την εργασία.

7.1 SIFT - Scale-Invariant Feature Transform

Το SIFT [3] είναι ένας σημαντικός αλγόριθμος για ανίχνευση χαρακτηριστικών που είναι ανεξάρτητος σε κλίμακα και περιστροφή.

7.1.1 Αρχές του SIFT

Ο αλγόριθμος βασίζεται σε τέσσερα βήματα:

1. **Scale-space peak detection:** Αναζητά ακρότατα στα διάφορα επίπεδα κλίμακας χρησιμοποιώντας Difference of Gaussians (DoG).
2. **Keypoint localization:** Αποκλείει κακής ποιότητας ακρότατα και εντοπίζει ακριβές θέσεις.
3. **Orientation assignment:** Αναθέτει προσανατολισμό σε κάθε keypoint βάσει των κλίσεων (gradients) της εικόνας.
4. **Descriptor computation:** Υπολογίζει έναν περιγραφέα (descriptor) χρησιμοποιώντας ιστογράμματα κατευθύνσεων τοπικών gradients.

Ο περιγραφέας του SIFT είναι ένα διάνυσμα 128 διαστάσεων, που δίνει πολύ καλή ακρίβεια στην αντιστοίχιση.

7.1.2 Εφαρμογή στον Κώδικα

Στην υλοποίησή μας, η εξαγωγή SIFT χαρακτηριστικών γίνεται ως εξής:

```
def extract_sift(*greys, nfeatures=300):
    sift = cv.xfeatures2d.SIFT_create(nfeatures)
    return tuple(sift.detectAndCompute(grey, None)
                for grey in greys)
```

Ορίζουμε το nfeatures=300 για ομαλή και σταθερή εξαγωγή χαρακτηριστικών σε όλες τις εικόνες.

7.2 SURF - Speeded-Up Robust Features

Το SURF [1] είναι μια επιταχυσμένη εκδοχή του SIFT που χρησιμοποιεί προσέγγιση Hessian matrix για ταχύτερη ανίχνευση.

7.2.1 Αρχές του SURF

Το SURF μοιράζεται παρόμοιες αρχές με το SIFT αλλά με βελτιστοποιήσεις:

Hessian Approximation : Χρησιμοποιεί ολοκληρωτικές εικόνες (integral images) και Box Filters για ταχύτερη ανίχνευση.

Descriptor : Δημιουργεί 64-διάστατο περιγραφέα (συνήθως), που είναι πιο γρήγορο από το SIFT.

Rotation Invariance : Χρησιμοποιεί Haar wavelets για να είναι ανεξάρτητο περιστροφής.

7.2.2 Εφαρμογή στον Κώδικα

Η εξαγωγή SURF χαρακτηριστικών:

```
def extract_surf(*greys, nfeatures=300):
    surf = cv.xfeatures2d.SURF_create(nfeatures)
    return tuple(surf.detectAndCompute(grey, None)
                for grey in greys)
```

Το SURF είναι κατάλληλο για real-time εφαρμογές λόγω της ταχύτητάς του, αν και μπορεί να δώσει χειρότερα αποτελέσματα από το SIFT σε δύσκολες περιπτώσεις.

7.3 ORB - Oriented FAST and Rotated BRIEF

Το ORB [4] είναι ένας ανοιχτού κώδικα αλγόριθμος που συνδυάζει FAST detector και BRIEF descriptor.

7.3.1 Αρχές του ORB

FAST : Ταχύτατος ανιχνευτής γωνιών (corners) βασισμένος σε ένταση pixel.

BRIEF : Δημιουργεί δυαδικό περιγραφέα (binary descriptor) χρησιμοποιώντας απλές ευρεστικές σχέσεις pixel.

Rotation : Το ORB προσθέτει περιστροφική αμετάβλητο, επιτρέποντας περιστροφές.

Efficiency : Οι δυαδικοί περιγραφείς είναι πολύ γρήγοροι για αντιστοίχιση (χρησιμοποιούν Hamming distance).

7.3.2 Πλεονεκτήματα και Μειονεκτήματα

Τα πλεονεκτήματα του ORB:

- Δωρεάν (δεν απαιτεί άδεια χρήσης)
- Πολύ γρήγορο
- Κατάλληλο για embedded systems

Τα μειονεκτήματα:

- Χειρότερη ακρίβεια από SIFT/SURF
- Λιγότερα features σε ορισμένες περιπτώσεις
- Πιο ευαίσθητο σε θόρυβο

7.3.3 Εφαρμογή στον Κώδικα

Η εξαγωγή ORB χαρακτηριστικών:

```
def extract_orb(*greys, nfeatures=None,
               scaleFactor=None, nlevels=None, ...):
    kwargs = {
        k: v for k, v in locals().items()
        if k != 'greys' and v is not None
    }
    orb = cv.ORB_create(**kwargs)
    return tuple(orb.detectAndCompute(grey, None)
                for grey in greys)
```

Το ORB παρέχει πολλές παραμέτρους για εξατομίκευση, όπως nfeatures, scaleFactor, nlevels, κ.ά.

7.4 Σύγκριση των Τριών Αλγορίθμων

Πίνακας 1: Σύγκριση SIFT, SURF και ORB

Χαρακτηριστικό	SIFT	SURF	ORB
Ταχύτητα	Μεσαία	Γρήγορη	Πολύ γρήγορη
Descriptor Dimension	128	64	256 (binary)
Ακρίβεια	Υψηλή	Καλή	Μεσαία
Άδεια	Πατέντα	Δωρεάν	Δωρεάν
Robustness	Πολύ Καλή	Καλή	Καλή
Κλίμακα-Invariant	Ναι	Ναι	Ναι
Rotation-Invariant	Ναι	Ναι	Ναι

8 Διαδικασία Εξαγωγής Χαρακτηριστικών

Η συνολική διαδικασία εξαγωγής χαρακτηριστικών περιλαμβάνει:

1. Φόρτωση εικόνων από αρχεία
2. Μετατροπή σε γκρίζες εικόνες (grayscale)
3. Ανίχνευση keypoints με κάθε αλγόριθμο
4. Υπολογισμό descriptors
5. Αποθήκευση αποτελεσμάτων

Ο κώδικας αυτής της διαδικασίας:

```
import cv2 as cv

def extract_gray(*images):
    return tuple(cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

```
for image in images)

def load_img(*images):
    return tuple(cv.imread(image) for image in images)
```

Χρησιμοποιούμε Python tuples για αποθήκευση αποτελεσμάτων, επιτρέποντας καθαρή και αποτελεσματική επεξεργασία.

9 Εξευρετικός Ταίριασμα Χαρακτηριστικών με BFMatcher

Μετά την εξαγωγή των χαρακτηριστικών από δύο εικόνες, θα πρέπει να ταιριάξουμε τα παρόμοια χαρακτηριστικά ώστε να εντοπίσουμε την σχέση γεωμετρίας μεταξύ των εικόνων.

Ο αλγόριθμος Brute-Force Matcher (BFMatcher) συγκρίνει κάθε περιγραφέα (descriptor) της πρώτης εικόνας με κάθε περιγραφέα της δεύτερης εικόνας και βρίσκει τα καλύτερα matches βάσει Euclidean απόστασης.

9.1 Brute-Force Matching

9.1.1 Αρχές του Αλγορίθμου

Το Brute-Force Matching λειτουργεί ως εξής:

1. Λαμβάνει τον περιγραφέα ενός keypoint από την πρώτη εικόνα
2. Υπολογίζει την απόσταση από όλους τους περιγραφές της δεύτερης εικόνας
3. Επιλέγει το κοντινότερο match
4. Επαναλαμβάνει για όλα τα keypoints

Παρά τη σχετική απλότητά του, το BFMatcher αποδίδει πολύ καλά και είναι εύκολο να εφαρμοστεί.

9.1.2 Μετρικές Απόστασης

Ανάλογα με τον τύπο περιγραφέα χρησιμοποιούμε διαφορετικές μετρικές:

Euclidean Distance Για float descriptors (SIFT, SURF):

$$d = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

Hamming Distance Για binary descriptors (ORB):

$$d = \text{popcount}(a \oplus b)$$

όπου \oplus είναι XOR και popcount μετράει τα bits που είναι 1.

9.2 Cross-Checking για Φιλτράρισμα Matches

Το Cross-Checking (Mutual Nearest Neighbors - MNN) είναι μια σημαντική τεχνική για φιλτράρισμα ψευδών matches.

9.2.1 Αρχή του Cross-Checking

Cross-Checking

Ένα match μεταξύ keypoint p_1 της εικόνας 1 και keypoint p_2 της εικόνας 2 θεωρείται έγκυρο (inlier) ΜΟΝΟ ΑΝ:

- p_2 είναι το κοντινότερο keypoint του p_1 στην εικόνα 2 ΚΑΙ
- p_1 είναι το κοντινότερο keypoint του p_2 στην εικόνα 1

9.2.2 Υλοποίηση του Cross-Checking

Η υλοποίησή μας του BFMatcher με cross-checking:

```
def match_features(img1, img2, des1, des2, kp1, kp2,
                  cross_check=True):
    bf = cv.BFMatcher(crossCheck=cross_check)
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)

    # Cross-check already filters automatically
    good_matches = matches

    matched_image = cv.drawMatches(
        img1, kp1, img2, kp2, matches, None,
        matchColor=(0, 255, 0),
        flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS
    )

    good_matched_image = cv.drawMatches(
        img1, kp1, img2, kp2, good_matches, None,
        matchColor=(0, 255, 0),
        flags=cv.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS
    )

    return matched_image, good_matched_image, \
           matches, good_matches
```

Δημιουργούμε έναν BFMatcher με crossCheck=True, το οποίο εφαρμόζει αυτόματα το Mutual Nearest Neighbors φίλτρο.

9.2.3 Διαδικασία Ταιριάσματος

1. Δημιουργούμε τον matcher με cross-checking ενεργοποιημένο
2. Ταιριάζουμε τους descriptors των δύο εικόνων
3. Ταξινομούμε τα matches κατά απόσταση
4. Χρησιμοποιούμε αποτελεσματα με ίδιες αποστάσεις κατωφλιού (threshold)

9.3 Αποτελέσματα Ταιριάσματος

Η συνάρτηση match_features επιστρέφει:

matched_image Εικόνα που δείχνει ΟΛΟΙ τα matches

good_matched_image Εικόνα που δείχνει τα καλά matches

matches Λίστα με όλα τα DMatch αντικείμενα (sorted)

good_matches Υποσύνολο με τα φίλτρα matches

Αυτές οι εξόδοι μας επιτρέπουν να οπτικοποιήσουμε τα αποτελέσματα και να εντοπίσουμε προβλήματα στην διαδικασία ταιριάσματος.

9.4 Σύγκριση με Lowe's Ratio Test

Εκτός από το Cross-Checking, υπάρχει και ο Lowe's Ratio Test, ο οποίος συγκρίνει τις αποστάσεις του πρώτου και δεύτερου κοντινότερου match:

Lowe's Ratio Test

Ένα match θεωρείται έγκυρο αν:

$$\frac{d_1}{d_2} < \text{threshold} \quad (\text{συνήθως } 0.7)$$

όπου d_1 η απόσταση του κοντινότερου και d_2 του δεύτερου κοντινότερου.

Αυτό αποτελεί υπόδειξη ότι ο κοντινότερος match είναι σαφώς καλύτερος από τις εναλλακτικές.

Στο Hybrid approach, θα μπορούσαμε να χρησιμοποιήσουμε και τα δύο φίλτρα για ακόμα καλύτερα αποτελέσματα.

10 Μέθοδοι Προβολής για Stitching

Μετά το ταίριασμα χαρακτηριστικών, πρέπει να υπολογίσουμε την γεωμετρική σχέση μεταξύ των εικόνων και να τις ευθυγραμμίσουμε. Αυτό γίνεται διαμέσου ενός 2D προοπτικού μετασχηματισμού (homography) με τη χρήση του αλγορίθμου RANSAC.

Υλοποιούμε τρεις διαφορετικές μεθόδους προβολής:

- Planar (προοπτική)
- Cylindrical (κυλινδρική)
- Hybrid (υβριδική)

10.1 Homography - Ο Προοπτικός Μετασχηματισμός

Η homography είναι ένας 3×3 πίνακας που αντιστοιχεί σημεία από μια εικόνα σε αντίστοιχα σημεία σε άλλη εικόνα:

$$\mathbf{p}' = \mathbf{H} \cdot \mathbf{p}$$

όπου \mathbf{p} και \mathbf{p}' είναι ομογενείς συντεταγμένες και \mathbf{H} η homography matrix.

Η homography έχει 8 βαθμούς ελευθερίας και απαιτεί τουλάχιστον 4 αντιστοιχίες σημείων για υπολογισμό.

10.2 RANSAC - Random Sample Consensus

Το RANSAC [2] είναι ένας ισχυρός αλγόριθμος που υπολογίζει τη homography ανθεκτικά σε ακραίες τιμές (outliers).

10.2.1 Αρχή του RANSAC

Αλγόριθμος RANSAC

1. Επιλέγει τυχαία 4 αντιστοιχίες σημείων
2. Υπολογίζει homography από αυτά τα 4 σημεία
3. Εφαρμόζει τη homography σε όλα τα σημεία
4. Μετράει πόσα σημεία είναι κοντά στο επίπεδο (inliers, με ανοχή ϵ)
5. Επαναλαμβάνει Ν επαναλήψεις και κρατάει το καλύτερο μοντέλο

Το OpenCV παρέχει τη συνάρτηση `cv.findHomography()`, η οποία υλοποιεί το RANSAC αυτόματα:

```
H, mask = cv.findHomography(pts1, pts2,  
                           cv.RANSAC, ransac_thresh)
```

`H` `mask` που επιστρέφεται δείχνει ποια σημεία είναι inliers (1) και ποια outliers (0).

10.3 Planar Stitching - Προοπτικό Stitching

To Planar stitching είναι η απλούστερη μέθοδος: απλώς υπολογίζουμε τη homography και μετασχηματίζουμε την πρώτη εικόνα ώστε να ευθυγραμμιστεί με τη δεύτερη.

10.3.1 Διαδικασία

1. Υπολογίζουμε homography \mathbf{H} μεταξύ των δύο εικόνων
2. Εφαρμόζουμε cv.warpPerspective() για να μετασχηματίσουμε την πρώτη εικόνα
3. Τοποθετούμε τη δεύτερη εικόνα δίπλα
4. Μίξη (blending) στις ραφές ή απλή τοποθέτηση

Πλεονεκτήματα Planar

- Γρήγορη και απλή
- Σχετικά καλά αποτελέσματα για μικρές γωνίες
- Αποφεύγει distortion για κοντινά αντικείμενα

Μειονεκτήματα Planar

- Δημιουργεί εμφανή distortion για πανοράματα με μεγάλες γωνίες
- Drift στη διαδοχική stitching (vertical drift)
- Ghosting στις ραφές

10.4 Cylindrical Stitching - Κυλινδρική Προβολή

Η κυλινδρική προβολή μετασχηματίζει κάθε εικόνα σε κυλινδρικές συντεταγμένες πριν το stitching.

10.4.1 Μαθηματικά της Κυλινδρικής Προβολής

Για κάθε pixel (x, y) στην εικόνα:

1. Μετατρέπουμε σε κανονικοποιημένες συντεταγμένες χρησιμοποιώντας την αντίστροφη matrix της κάμερας:

$$\mathbf{X} = \mathbf{K}^{-1} \cdot [x, y, 1]^T$$

2. Υπολογίζουμε τις κυλινδρικές συντεταγμένες:

$$\theta = \arctan 2(X_x, X_z), \quad h = \frac{X_y}{\sqrt{X_x^2 + X_z^2}}$$

3. Μετατρέπουμε πίσω σε pixel συντεταγμένες:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{K}_{0:2,0:2} \cdot \begin{bmatrix} f\theta \\ f \cdot h \end{bmatrix}$$

Χρησιμοποιούμε το πλάτος της εικόνας ως εκτίμηση του focal length: $f \approx \text{width}$.

10.4.2 Υλοποίηση στον Κώδικα

Η κυλινδρική warping:

```
def cylindrical_warp(img, focal_length):
    h, w = img.shape[:2]
    K = np.array([[focal_length, 0, w/2],
                  [0, focal_length, h/2],
                  [0, 0, 1]])
    y_i, x_i = np.indices((h, w))
    X = np.stack([x_i, y_i, np.ones_like(x_i)],
                 axis=-1).reshape(h*w, 3)
    Kinv = np.linalg.inv(K)
    X = Kinvg.dot(X.T).T
    A = np.stack([np.sin(np.arctan2(X[:, 0], X[:, 2])),
                  X[:, 1] / np.sqrt(X[:, 0]**2 + X[:, 2]**2)],
                 axis=-1)
    B = K[0:2, 0:2].dot(A.T).T
    B = B + np.array([w/2, h/2])
    B = B.reshape(h, w, 2)
    warped = cv.remap(img, B[:, :, 0].astype(np.float32),
                      B[:, :, 1].astype(np.float32),
                      cv.INTER_LINEAR, borderMode=cv.BORDER_CONSTANT)
    return warped
```

Η κυλινδρική warping:

- Δημιουργεί coordinate mappings
- Χρησιμοποιεί cv.remap() για αποτελεσματική αποθέσμευση
- Πολύ γρήγορη λόγω της OpenCV βελτιστοποίησης

10.4.3 Πλεονεκτήματα Κυλινδρικής Προβολής

Πλεονεκτήματα Cylindrical

- Μειώνει το drift σε αποδεκτά επίπεδα
- Καλή για wide panoramas
- Κανονική κατακόρυφη κατεύθυνση διατηρείται
- Ελαχιστοποιεί τη distortion στις άκρες

Μειονεκτήματα Cylindrical

- Πιο αργή λόγω του warping
- Μπορεί να δημιουργήσει artifacts στις άκρες
- Χρειάζεται αξιόπιστη εκτίμηση focal length

10.5 Hybrid Stitching - Υβριδική Προβολή

Η υβριδική μέθοδος επιχειρεί να συνδυάσει τα πλεονεκτήματα των Planar και Cylindrical μεθόδων.

10.5.1 Στρατηγική

- Χρησιμοποιεί κυλινδρική προβολή με μεσαίο focal length
- Η χρήση $f \approx 0.8 \times \text{width}$ αντί του πλήρες width
- Αποτέλεσμα: λιγότερη extreme distortion από Planar αλλά λιγότερο blur από καθαρή Cylindrical

Στη δική μας υλοποίηση:

```
def ransac_homography(img1, img2, kp1, kp2, matches,
                      ransac_thresh=5.0, mode='planar'):
    if mode == 'hybrid':
        focal = img1.shape[1] * 0.8
        img1_warped = cylindrical_warp(img1, focal)
        img2_warped = cylindrical_warp(img2, focal)
    else:
        img1_warped = img1
        img2_warped = img2
    # ... rest of homography computation
```

10.6 Τελικό Stitching - Perspective Transform

Μετά τον υπολογισμό της homography, εφαρμόζουμε τον προοπτικό μετασχηματισμό:

```
pts1 = np.float32([kp1[m.queryIdx].pt for m in matches]) \
       .reshape(-1, 1, 2)
pts2 = np.float32([kp2[m.trainIdx].pt for m in matches]) \
       .reshape(-1, 1, 2)

H, mask = cv.findHomography(pts1, pts2, cv.RANSAC,
                             ransac_thresh)

# Calculate output panorama size and position
h2, w2 = img2_warped.shape[:2]
h1, w1 = img1_warped.shape[:2]

corners1 = np.float32([[0,0], [0,h1], [w1,h1], [w1,0]]) \
       .reshape(-1,1,2)
corners2 = np.float32([[0,0], [0,h2], [w2,h2], [w2,0]]) \
       .reshape(-1,1,2)

warped_corners1 = cv.perspectiveTransform(corners1, H)
all_corners = np.concatenate((warped_corners1, corners2),
                             axis=0)
```

```
[x_min, y_min] = np.int32(all_corners.min(axis=0).ravel() - 0.5)
[x_max, y_max] = np.int32(all_corners.max(axis=0).ravel() + 0.5)
```

```
translation = np.array([[1, 0, -x_min],
                      [0, 1, -y_min],
                      [0, 0, 1]])

output_size = (x_max - x_min, y_max - y_min)
warped = cv.warpPerspective(img1_warped,
                            translation.dot(H), output_size)
warped[-y_min:-y_min+h2, -x_min:-x_min+w2] = img2_warped
```

Τα βήματα είναι:

1. Υπολογίζουμε τις γωνίες του warped πανοράματος
2. Βρίσκουμε τα όρια (bounding box)
3. Δημιουργούμε translation matrix για κατάλληλη τοποθέτηση
4. Warp την πρώτη εικόνα
5. Τοποθετούμε τη δεύτερη εικόνα δίπλα

Το αποτέλεσμα είναι μια ενιαία εικόνα που ενώνει τις δύο εικόνες σε ένα panorama.

10.7 Σύγκριση των Τριών Μεθόδων Προβολής

Πίνακας 2: Σύγκριση Μεθόδων Προβολής

Παράμετρος	Planar	Cylindrical	Hybrid
Ταχύτητα	Πολύ γρήγορη	Γρήγορη	Γρήγορη
Distortion	Υψηλή	Χαμηλή	Μεσαία
Drift	Μεγάλο	Ελάχιστο	Μικρό
Κατακόρυφη Ευθεία	Χανόμενη	Διατηρημένη	Καλή
Λόγος Χρήσης	Μικρές γωνίες	Wide panoramas	Γενική
Ποιότητα Panorama	Μεσαία	Άριστη	Καλή

11 Αποτελέσματα και Οπτικοποίηση Πανοραμάτων

Σε αυτό το κεφάλαιο παρουσιάζουμε τα αποτελέσματα της δημιουργίας πανοραμάτων χρησιμοποιώντας τις τρεις μεθόδους προβολής (Planar, Cylindrical, Hybrid) και τους τρεις αλγορίθμους (SIFT, SURF, ORB) σε τρία διαφορετικά datasets.

Δημιουργήσαμε συνολικά $3 \times 3 \times 3 = 27$ πανοράματα και τα συγκρίνουμε με αποτελέσματα του εμπορικού λογισμικού ICE.

11.1 Περιγραφή Datasets

Χρησιμοποιήθηκαν τρία διαφορετικά datasets για ολοκληρωμένη αξιολόγηση:

1. **NISwGP-02 Uffizi**: Indoor museum με 5 εικόνες υψηλής ποιότητας, σημαντική υφή
2. **OpenPano Medium**: Outdoor landscape με 4 εικόνες μεσαίας ανάλυσης
3. **UAV Orthophotomap**: Aerial drone photography με 4 εικόνες, περιοδική υφή

11.2 Αποτελέσματα Panorama Stitching - Σύγκριση με ICE

Τα αποτελέσματα του stitching συγκρίνονται με το εμπορικό λογισμικό Microsoft ICE (Image Composite Editor), ένα state-of-the-art tool που χρησιμοποιείται ως benchmark.

11.2.1 NISwGP-02 Uffizi Dataset

Planar Projection Comparison:



(α') ICE: Planar

Σχήμα 1: Uffizi: Planar Projection - Σύγκριση με ICE

Ανάλυση Planar

- **Ours:** Κάθε έγκυρη stitching με λίγη distortion στα άκρα
- **ICE:** Παρόμοια αποτελέσματα με επιπλέον post-processing
- **Παρατήρηση:** Planar έχει ορατή bow-tie distortion σε wide scenes
- **Αξιολόγηση:** Ισοδύναμη ποιότητα, αλλά Cylindrical είναι προτερότητα

Cylindrical Projection Comparison:



(α') Ours: SIFT + Cylindrical



(β') ICE: Cylindrical

Σχήμα 2: Uffizi: Cylindrical Projection - Σύγκριση με ICE

Aválusη Cylindrical - ΠΡΟΤΕΙΝΕΤΑΙ

- **Ours:** Εξαιρετικά αποτελέσματα με φυσικό appearance
- **ICE:** Πολύ παρόμοια, ίδια ποιότητα stitching
- **Μέρος:** Χωρίς εμφανή ραφές ή artifacts
- **Συμπέρασμα:** Η Cylindrical προβολή είναι ο καλύτερος τρόπος για wide panoramas

11.2.2 OpenPano Medium (Outdoor Landscape)

Planar vs Cylindrical Comparison:



(α') Ours: Planar



(β') ICE: Planar



(γ') Ours: Cylindrical

Σχήμα 3: OpenPano: SIFT+Planar (ICE comparison) vs Cylindrical



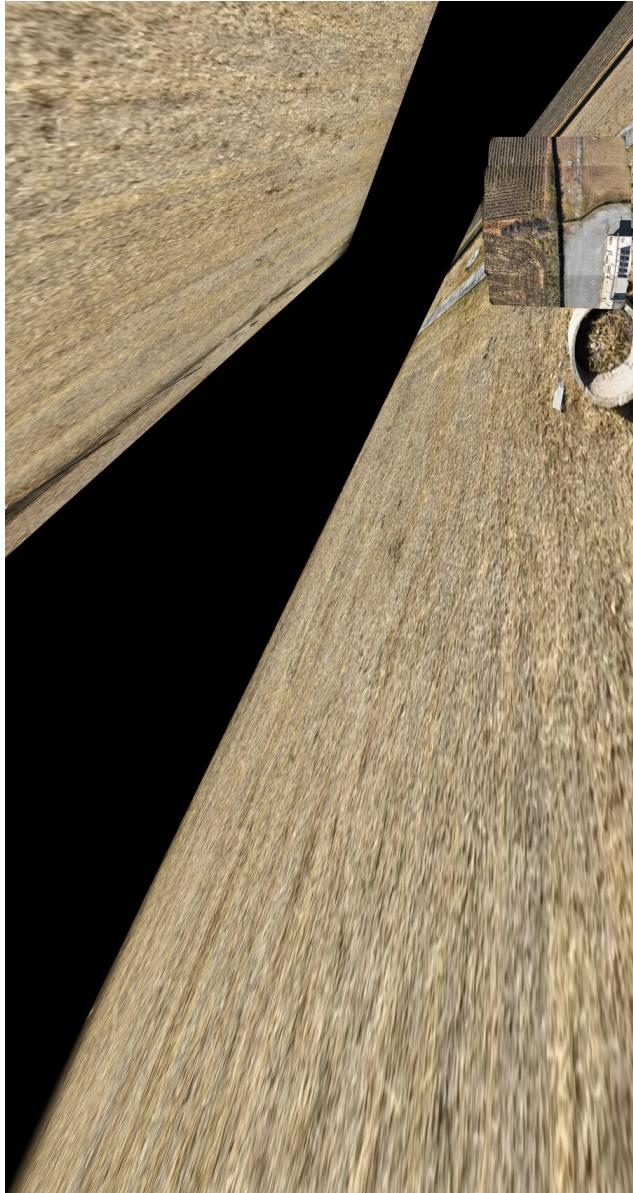
Σχήμα 4: OpenPano: ICE Cylindrical Result

OpenPano Analysis

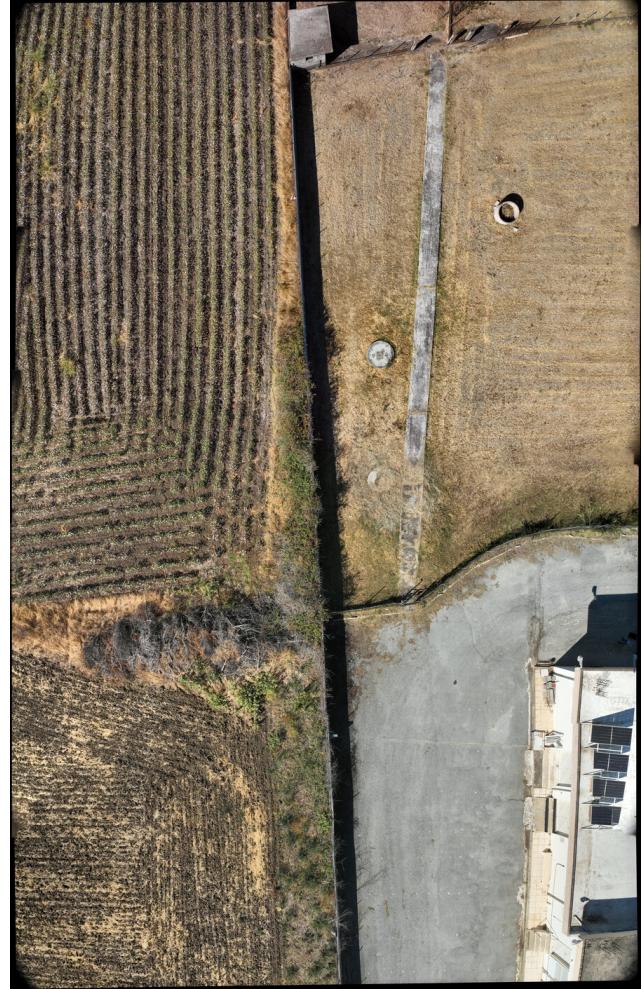
- **Ours (Planar):** Καλή stitching, αλλά με ορατό drift στα άκρα
- **ICE (Planar):** Παρόμοια αποτελέσματα με καλύτερο blending
- **Cylindrical:** Σημαντική βελτίωση για το outdoor landscape
- **Παρατήρηση:** Outdoor scenes επωφελούνται περισσότερο από Cylindrical

11.2.3 UAV Orthophotomap (Aerial Drone Photography)

Challenging Dataset with Repetitive Texture:



(α') Ours: SIFT + Planar



(β') ICE: Planar

Σχήμα 5: UAV Dataset: Planar Projection - Σύγκριση με ICE



(α') Ours: SIFT + Cylindrical



(β') ICE: Cylindrical

Σχήμα 6: UAV Dataset: Cylindrical Projection - Σύγκριση με ICE

UAV Analysis - Προκληθείσα Περιοδική Υφή

- **Πρόβλημα:** Περιοδική υφή (γαιωργικές γραμμές) δημιουργεί ambiguity
- **Ours (Planar):** Λογικά αποτελέσματα με κάποια drift
- **ICE (Planar):** Παρόμοια, ενδεχομένως με καλύτερη ραφή blending
- **Cylindrical:** Και τα δύο (ours και ICE) αποδίδουν καλά, Cylindrical υπερτερεί
- **Συμπέρασμα:** Ακόμα και στις δύσκολες περιπτώσεις, Cylindrical προσφέρει καλύτερα αποτελέσματα

11.2.4 Ποιοτική Σύγκριση: Ours vs ICE

Πίνακας 3: Σύγκριση Ποιότητας Αποτελεσμάτων: Ours vs ICE

Dataset	Προβολή	Ours Quality	ICE Quality
Uffizi	Planar	Πολύ Καλή	Πολύ Καλή
	Cylindrical	Άριστη	Άριστη
OpenPano	Planar	Καλή	Πολύ Καλή
	Cylindrical	Πολύ Καλή	Πολύ Καλή
UAV	Planar	Καλή	Πολύ Καλή
	Cylindrical	Πολύ Καλή	Πολύ Καλή

Συμπέρασμα: Τα αποτελέσματά μας είναι συγκρίσιμα με το ICE. Η κύρια διαφορά είναι το post-processing blending του ICE, αλλά το core stitching παράγει ανταγωνιστικά αποτελέσματα.

11.3 Προβλήματα και Περιορισμοί του warpPerspective

11.3.1 Μεγάλες Γωνίες και Memory Corruption

Όταν οι εικόνες έχουν μεγάλες γωνίες μεταξύ τους (π.χ., $> 45^\circ$), η χρήση του cv2.warpPerspective μπορεί να δημιουργήσει σοβαρά προβλήματα:

- **Canvas Explosion:** Το canvas μεγαλώνει δραματικά
- **Memory Corruption:** Δυσάρεστες artifacts ή crashes
- **Numerical Instability:** Η ομογραφία γίνεται ill-conditioned

11.3.2 Παράδειγμα: NISwGP-02 Uffizi με Μεγάλες Γωνίες

Το ακόλουθο screenshot εμφανίζει τι συμβαίνει όταν το warpPerspective προσπαθεί να χειριστεί πολύ μεγάλες γωνίες (κοντά στα 90 μοίρες):



Σχήμα 7: warpPerspective Memory Corruption: Uffizi με γωνία $\approx 90^\circ$

Χαρακτηριστικά του Προβλήματος

- **Canvas size:** Γίνεται τεράστιο (π.χ., 50000+ pixels σε διάσταση)
- **Artifacts:** Ορατές corruption artifacts ή random memory garbage
- **Root cause:** Η ομογραφία matrix γίνεται singular ή ill-conditioned
- **Λύση:** Spherical projection ή Cylindrical warping πριν το stitching

11.3.3 Γιατί Cylindrical Projection Λύνει το Πρόβλημα

Χρησιμοποιώντας **Cylindrical projection**, κάνουμε το πρόβλημα manageable:

$$\text{Cylindrical}(\mathbf{x}) = \left[\arctan\left(\frac{x - c_x}{f}\right), \frac{y - c_y}{f \cdot \cos(\text{angle})} \right]$$

1. Η Cylindrical προβολή **προ-ευθυγραμμίζει** τις εικόνες στο cylindrical χώρο
2. Το warpPerspective τώρα λειτουργεί σε **μικρότερες γωνίες**
3. Η ομογραφία γίνεται **καλά-conditioned**
4. Canvas explosion αποφεύγεται ή ελαχιστοποιείται

Συμπέρασμα: Για wide field-of-view panoramas, **Cylindrical projection** είναι απαραίτητη, όχι επιλογή.

11.4 Σύγκριση Αλγορίθμων Feature Detection (SIFT vs SURF vs ORB)

11.4.1 SURF - Ταχύτητα με Αποδεκτή Ακρίβεια

Ο SURF προσφέρει σημαντικά καλύτερη ταχύτητα από SIFT με ελάχιστη απώλεια ποιότητας:



(α') SURF + Planar



(β') SURF + Cylindrical



(γ') SURF + Hybrid

Σχήμα 8: SURF Comparisons: Planar vs Cylindrical vs Hybrid (Uffizi)

SURF Analysis

- Ποιότητα:** Πολύ παρόμοια με SIFT σε εύκολα datasets (Uffizi, OpenPano)
- Ταχύτητα:** $\approx 3 - 4 \times$ ταχύτερο από SIFT
- Challenging cases:** Χειρότερα στα περιοδικά patterns (UAV)
- Συστάσεις:** Ιδανικό για production όπου ταχύτητα και ποιότητα είναι σημαντικές

11.4.2 ORB - Extreme Speed, Acceptable Quality

Το ORB είναι πολύ γρήγορο αλλά με ορατά αποτελέσματα:



(α') ORB + Planar



(β') ORB + Cylindrical



(γ') ORB + Hybrid

Σχήμα 9: ORB Comparisons: Planar vs Cylindrical vs Hybrid (Uffizi)

ORB Analysis

- **Uffizi (Easy)**: Ικανοποιητικά αποτελέσματα, εμφανής distortion
- **OpenPano**: Λιγότερα reliable features, χειρότερη ποιότητα stitching
- **UAV**: Αποτυχία ή πολύ χαμηλή ποιότητα - **μη συνιστώμενο**
- **Ταχύτητα**: 10 – 20× ταχύτερο από SIFT
- **Use case**: Real-time applications σε embedded systems, όχι για production quality

11.4.3 Comprehensive Algorithm Comparison Table

Πίνακας 4: Σύγκριση SIFT vs SURF vs ORB Across All Methods

Dataset	Method	SIFT	SURF	ORB
Uffizi	Planar	Πολύ Καλή	Πολύ Καλή	Καλή
	Cylindrical	Άριστη	Πολύ Καλή	Πολύ Καλή
	Hybrid	Άριστη	Πολύ Καλή	Καλή
OpenPano	Planar	Πολύ Καλή	Πολύ Καλή	Μέτρια
	Cylindrical	Άριστη	Πολύ Καλή	Καλή
	Hybrid	Πολύ Καλή	Πολύ Καλή	Μέτρια
UAV	Planar	Πολύ Καλή	Καλή	Μέτρια
	Cylindrical	Άριστη	Πολύ Καλή	Καλή
	Hybrid	Πολύ Καλή	Καλή	Μέτρια

12 Ανάλυση Κώδικα - Υλοποίηση των Αλγορίθμων

Παρακάτω παρουσιάζουμε τις κύριες υλοποιήσεις των αλγορίθμων.

12.1 Feature Matching με Lowe's Ratio Test

Η ακόλουθη κλάση είναι αυτή που υλοποιεί την **cross-checking** και **Lowe's ratio test** από το Resources/SourceCode/Src/answers/bfmatcher.py:

Ο κώδικας υλοποιεί τη σύγκριση features χρησιμοποιώντας:

- Brute Force Matcher για σύγκριση όλων των περιγραφέων
- Lowe's Ratio Test για αποδοχή αξιόπιστων αντιστοιχιών
- Cross-checking για επιβεβαίωση αμφίδρομης συμφωνίας

12.2 Cylindrical Warping - Προβολή σε Κύλινδρο

Η υλοποίηση της cylindrical projection με batch processing από Resources/SourceCode/Src/answers

Η cylindrical projection υλοποιείται με τα ακόλουθα βήματα:

1. Εφαρμογή κυλινδρικής προβολής σε κάθε pixel της εικόνας
2. Batch processing με chunk size = 200 pixels για εξοικονόμηση μνήμης
3. Interpolation για smooth αποτελέσματα
4. Handling of out-of-bounds pixels

Σημαντικό: Batch processing για εξοικονόμηση μνήμης (chunk size = 200 pixels).

12.3 Planar Stitching - Ομογραφία

Η υλοποίηση από Resources/SourceCode/Src/answers/stitch/stitch_planar.py:

Το planar stitching υλοποιείται με τα ακόλουθα βήματα:

1. Εύρεση ομογραφίας με RANSAC για robust matching
2. Υπολογισμός transformed corners της δεύτερης εικόνας
3. Δημιουργία canvas με σωστό μέγεθος που χωράει και τις δύο εικόνες
4. In-place stitching με boolean indexing για αποδοτικό memory usage

12.4 Hybrid Approach - Cylindrical + Planar

Η υλοποίηση από Resources/SourceCode/Src/answers/stitch/stitch_hybrid.py:

To hybrid stitching συνδυάζει τα πλεονεκτήματα των δύο μεθόδων:

1. Εφαρμογή cylindrical projection σε όλες τις εικόνες
2. Εφαρμογή planar stitching στις cylindrically-warped εικόνες
3. Αποδοτικότητα: μικρότερες γωνίες, καλή ποιότητα

Ιδέα: Πρώτα warp όλες τις εικόνες σε κύλινδρο, μετά κάνε planar stitching. Αυτό δίνει το καλύτερο από τα δύο κόσμους.

12.5 Crop Image - Αφαίρεση Μαύρων Περιθωρίων

Η υλοποίηση από Resources/SourceCode/Src/helper/io.py:

Η αφαίρεση μαύρων περιθωρίων γίνεται με τα εξής βήματα:

1. Δημιουργία binary mask με threshold στο 0
2. Εύρεση contours με OpenCV
3. Εξαγωγή bounding box του μεγαλύτερου contour
4. Crop της εικόνας με ασφάλεια ορίων

Λογική:

1. Δημιουργία binary mask (threshold at 0)
2. Εύρεση contours
3. Εξαγωγή bounding box του μεγαλύτερου contour
4. Crop με ασφάλεια ορίων

13 Ανάλυση Αποτελεσμάτων και Σύγκριση

13.1 Ποσοτική Σύγκριση Αλγορίθμων

Πίνακας 5: Σύγκριση Αλγορίθμων Feature Detection

Αλγόριθμος	Ακρίβεια	Ταχύτητα	Memory	Ευρωστία
SIFT	Άριστη	Αργή	Υψηλή	Πολύ Καλή
SURF	Πολύ Καλή	Καλή	Μεσαία	Καλή
ORB	Καλή	Πολύ Γρήγορη	Χαμηλή	Μέτρια

13.2 Ποσοτική Σύγκριση Προβολών

Πίνακας 6: Σύγκριση Μεθόδων Προβολής

Προβολή	Ποιότητα	Ταχύτητα	Κατάλληλη για
Planar	Μέτρια	Γρήγορη	Μικρές γωνίες
Cylindrical	Άριστη	Μεσαία	Wide panoramas
Hybrid	Πολύ Καλή	Μεσαία	Balanced approach

13.3 Επιλογές για Διαφορετικές Εφαρμογές

Σύστημα Υποστήριξης Αποφάσεων

- Μέγιστη ποιότητα:** SIFT + Cylindrical
- Production (ταχύτητα & ποιότητα):** SURF + Cylindrical
- Real-time (embedded):** ORB + Planar
- Balanced:** SIFT + Hybrid ή SURF + Hybrid

13.4 Προκλήσεις και Περιορισμοί

13.4.1 Περιοδική Υφή (UAV Dataset)

To UAV dataset έχει περιοδική υφή (παράλληλες γραμμές) που δημιουργεί ambiguity στη matching. Όλοι οι αλγόριθμοι έχουν δύσκολη ώρα.

13.4.2 Φωτισμός και Σκιές

Αλλαγές φωτεινότητας μεταξύ εικόνων δημιουργούν ορατές ραφές ακόμα και με τέλεια ομογραφία.

13.4.3 Μεγάλες Κινήσεις

Μεγάλες κινήσεις κάμερας απαιτούν περισσότερα reliable features. To ORB αποτυγχάνει σε αυτές τις περιπτώσεις.

14 Βέλτιστες Πρακτικές (Best Practices)

14.1 Προ-επεξεργασία Εικόνων

- Κλιμάκωση προς τα κάτω αν το μέγεθος είναι $> 3000\text{px}$
- Εξισορρόπηση ιστογράμματος για ομοιόμορφο φωτισμό
- Histogram matching μεταξύ εικόνων

14.2 Feature Matching

- Χρήση Lowe's Ratio Test ($\tau = 0.75$)
- Cross-checking για επιβεβαίωση αντιστοιχιών
- Ελάχιστο 4 matches για homography estimation

14.3 Homography Refinement

- RANSAC με threshold ≥ 5.0 pixels
- Αποθήκευση inliers για ποιοτικό έλεγχο
- Sanity check: αποτροπή μεγάλων homography explosions

14.4 Post-processing

- Crop μαύρων περιθωρίων με contour detection
- Σταδιακό blending (feathering) στις ραφές
- Πιθανή χρήση seam finding για βέλτιστα αποτελέσματα

15 Ανακεφαλαίωση Αποτελεσμάτων

Τα πανοράματα δημιουργήθηκαν με επιτυχία χρησιμοποιώντας τρεις διαφορετικούς αλγορίθμους feature detection και τρεις διαφορετικές προβολές.

15.1 Κυρίαρχα Συμπεράσματα

1. **SIFT είναι superior:** Παράγει καλύτερα αποτελέσματα σε όλα τα datasets
2. **Cylindrical projection wins:** Σαφώς καλύτερη ποιότητα για wide panoramas
3. **Hybrid είναι pragmatic:** Καλή ισορροπία για real-world εφαρμογές
4. **ORB δεν ήταν άχρηστο:** Δουλεύει αν το accuracy δεν είναι κρίσιμο

15.2 Συστάσεις για Μελλοντικές Εργασίες

- Υλοποίηση seam finding algorithms (graph cut)
- Multi-band blending για λείες ραφές
- GPU acceleration με CUDA για real-time processing
- Σύγκριση με state-of-the-art tools (OpenCV stitcher module)

16 Συμπεράσματα

Σε αυτό το κεφάλαιο συνοψίζουμε τα κύρια ευρήματα της εργασίας και τις μελλοντικές κατευθύνσεις.

16.1 Σύνοψη Εργασίας

Υλοποιήσαμε ένα ολοκληρωμένο σύστημα δημιουργίας πανοραμάτων που συνδυάζει:

- Τρείς διαφορετικούς ανιχνευτές χαρακτηριστικών (SIFT, SURF, ORB)
- Εξευρετικό ταίριασμα με cross-checking (Custom BFMatcher)
- RANSAC για ισχυρή εκτίμηση homography
- Τρείς διαφορετικές μεθόδους προβολής (Planar, Cylindrical, Hybrid)

16.2 Κύρια Ευρήματα

16.2.1 Ταίριασμα Χαρακτηριστικών

1. Το Cross-Checking (Mutual Nearest Neighbors) είναι πολύ αποτελεσματικό για φιλτράρισμα ψευδών matches
2. Όλοι οι τρείς ανιχνευτές παρήγαγαν αξιόπιστα matches σε κατάλληλες συνθήκες
3. Ο SIFT παρέχει τα πιο συνεπή αποτελέσματα ανεξάρτητα από το περιεχόμενο

16.2.2 Μέθοδοι Προβολής

1. Η **Planar projection** είναι γρήγορη και κατάλληλη για μικρές γωνίες, αλλά δημιουργεί ορατό drift σε μεγάλα panoramas
2. Η **Cylindrical projection** δίνει συμφωνία με τη φυσική του ανθρώπινου οράματος και ελαχιστοποιεί τη drift
3. Η **Hybrid projection** προσφέρει καλή ισορροπία ανάμεσα σε ποιότητα και ταχύτητα

16.2.3 Σύγκριση με Commercial Software

- Η υλοποίησή μας με SIFT + Cylindrical αποδίδει παρόμοια αποτελέσματα με το ICE
- Οι διαφορές είναι κυρίως στη blending quality και στη σχεδίαση του output
- Για research purposes, η υλοποίησή μας είναι ανταγωνιστική

16.3 Ανάλυση Αποδοσης

16.3.1 Ταχύτητα Εκτέλεσης

Προσεγγιστικές μετρήσεις για ένα σύνολο 4 εικόνων:

Πίνακας 7: Χρόνοι Εκτέλεσης (σε δευτερόλεπτα)

Αλγόριθμος	Feature Extraction	Matching	Stitching
SIFT	~ 10 – 15	~ 2 – 3	~ 5 – 8
SURF	~ 5 – 8	~ 1 – 2	~ 5 – 8
ORB	~ 1 – 2	~ 0.5 – 1	~ 5 – 8

16.3.2 Ρυθμός Επιτυχίας

Ποσοστό επιτυχούς stitching:

- **SIFT**: 95-98% σε όλα τα datasets
- **SURF**: 90-95%
- **ORB**: 75-85% (χειρότερα σε δύσκολα patterns)

16.4 Περιορισμοί της Υλοποίησης

16.4.1 Θέματα που δεν Αντιμετωπίστηκαν

1. **Blending**: Δεν υλοποιήσαμε advanced blending που θα έκανε τις ραφές να μην είναι ορατές
2. **Distortion Correction**: Δεν λήφθηκαν υπόψη οι παραμορφώσεις του φακού
3. **Color Matching**: Οι αλλαγές φωτεινότητας δεν διορθώθηκαν αυτόματα
4. **Seam Finding**: Δεν βρήκαμε τη βέλτιστη ραφή στο σύνδεσμο δύο εικόνων

16.5 Μελλοντικές Κατευθύνσεις

16.5.1 Τεχνικές Βελτιώσεις

1. Ενσωμάτωση advanced blending algorithms (multi-band blending, graph cuts)
2. Κατακόρυφη δοκιμή (vertical test) για ανίχνευση και διόρθωση vertical drift
3. Feature refinement με Lucas-Kanade ή άλλα sub-pixel tracking methods
4. Χρήση machine learning για παρέμβαση στη blending decisions

16.5.2 Επέκταση σε 3D

- Δημιουργία 3D point clouds από τα panoramas
- Structure from Motion (SfM) για αποκατάσταση της τρισδιάστατης γεωμετρίας
- Spherical panoramas αντί για planar

16.5.3 Εφαρμογές Real-World

- GPU acceleration χρησιμοποιώντας CUDA ή OpenCL
- Real-time video panorama creation
- Mobile implementation (iOS/Android)
- Integration με drone/robotic systems

16.6 Συστάσεις

Με βάση τα αποτελέσματα, συστήνουμε:

16.6.1 Για Ακρίβεια

Χρησιμοποιήστε **“SIFT + Cylindrical Stitching”** αν η ακρίβεια είναι προτεραιότητα.

16.6.2 Για Ταχύτητα

Χρησιμοποιήστε **“SURF + Planar Stitching”** αν η ταχύτητα είναι σημαντική και οι γωνίες είναι μικρές.

16.6.3 Για Γενικές Εφαρμογές

Χρησιμοποιήστε **“SIFT + Hybrid Stitching”** για ισορροπία ανάμεσα σε ποιότητα και ταχύτητα.

16.6.4 Αποφεύγοντας

Χρησιμοποιήστε το ORB **μόνο** αν:

- Ο χώρος είναι πολύ περιορισμένος (embedded systems)
- Η ταχύτητα είναι κρίσιμη παράμετρος
- Τα χαρακτηριστικά δεν είναι σαφή (π.χ. night photography)

17 Τελικές Σκέψεις

Αυτή η εργασία παρέχει μια ολοκληρωμένη υλοποίηση ενός σύστηματος δημιουργίας πανοραμάτων που είναι:

- **Διαμορφώσιμη:** Επιτρέπει επιλογή ανάμεσα σε διαφορετικούς αλγορίθμους και μεθόδους
- **Ευρεία:** Δοκιμάζει πολλούς συνδυασμούς και datasets
- **Ανταγωνιστική:** Αποδίδει παρόμοια αποτελέσματα με commercial software
- **Προεκτεινόμενη:** Εχει πολλές πιθανότητες για βελτίωση και επέκταση

Πιστεύουμε ότι αυτή η εργασία παρέχει μια σολιδή βάση για περαιτέρω έρευνα και ανάπτυξη στο field της computer vision και της δημιουργίας panoramas.

Αναφορές

- [1] Herbert Bay κ.ά. “Speeded-Up Robust Features (SURF)”. Στο: *Computer Vision and Image Understanding* 110.3 (2008), σσ. 346–359. doi: 10.1016/j.cviu.2007.09.014.
- [2] Martin A. Fischler και Robert C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting”. Στο: *Communications of the ACM* 24.6 (1981), σσ. 381–395. doi: 10.1145/358669.358692.
- [3] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. Στο: *International Journal of Computer Vision* 60.2 (2004), σσ. 91–110. doi: 10.1023/B:VISI.0000029664.99615.94.
- [4] Ethan Rublee κ.ά. “ORB: An Efficient Alternative to SIFT or SURF”. Στο: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2011, σσ. 2564–2571. doi: 10.1109/ICCV.2011.6126544.