ΕΠΙΛΕΓΜΕΝΕΣ ΠΤΥΧΙΑΚΕΣ ΚΑΙ ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

STUDBOOK

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

ΤΟΜΟΣ 22
2026

Εκδίδεται μία φορά το χρόνο από το:

Τμήμα Πληροφορικής και Τηλεπικοινωνιών,
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών,
Πανεπιστημιούπολη, 15784 Αθήνα

Επιμέλεια έκδοσης:

Χάρης Θεοχάρης (Καθηγητής), Υπεύθυνος Έκδοσης
Λήδα Χαλάτση (ΕΤΕΠ), Σύνταξη & Γραφιστική Επιμέλεια

Εικόνα εξωφύλλου και εσωφύλλων:

Refik Anadol, Sample data visualisations of Refik Anadol's Unsupervised — Machine Hallucinations — MoMA — Fluid Dreams (2022) (Image credit: The Museum of Modern Art, New York, © Refik Anadol Studio)

Περιεχόμενα

Πρόλογος

Ο τόμος αυτός περιλαμβάνει περιλήψεις επιλεγμένων διπλωματικών και πτυχιακών εργασιών που εκπονήθηκαν στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών κατά το διάστημα **01/01/2025 - 31/12/2025**. Πρόκειται για τον **22ο τόμο** στη σειρά αυτή. Στόχος του θεσμού είναι η ενθάρρυνση της δημιουργικής προσπάθειας και η προβολή των πρωτότυπων εργασιών των φοιτητών του Τμήματος.

Η έκδοση αυτή είναι ψηφιακή, έχει δικό της ISSN και αναρτάται στην επίσημη ιστοσελίδα του Τμήματος ώστε να έχει μεγάλη προσβασιμότητα. Για το στόχο αυτό, σημαντική ήταν η συμβολή της Λήδας Χαλάτση που επιμελήθηκε και φέτος την ψηφιακή έκδοση και πέτυχε μια ελκυστική ποιότητα παρουσίασης, ενώ βελτίωσε και την ομοιογένεια των κειμένων.

Θα θέλαμε να ευχαριστήσουμε τους φοιτητές για το χρόνο που αφιέρωσαν για να παρουσιάσουν τη δουλειά τους στα πλαίσια αυτού του θεσμού και να τους συγχαρούμε για την ποιότητα των εργασιών τους. Ελπίζουμε η διαδικασία αυτή να προσέφερε και στους ίδιους μια εμπειρία που θα τους βοηθήσει στη συνέχεια των σπουδών τους ή της επαγγελματικής τους σταδιοδρομίας.

Αθήνα, Ιούλιος 2026

Πτυχιακές Εργασίες

Μέγιστη Συσκευασία Πολυγώνων μέσω Κινητού Ορίου Τοποθέτησης και Στοχαστικών Μεθόδων

Αλέξανδρος Ούλμαν Δερνιτσιώτης

ΠΕΡΙΛΗΨΗ

Στο πλαίσιο του διαγωνισμού CG:SHOP 2024, μελετήσαμε το πρόβλημα της μέγιστης συσκευασίας πολυγώνων. Στόχος ήταν η επιλογή και τοποθέτηση ενός υποσυνόλου πολυγώνων με προκαθορισμένες αξίες εντός κυρτής περιοχής, ώστε να μεγιστοποιείται το συνολικό άθροισμα των αξιών τους.

Η προσέγγισή μας βασίστηκε σε στοχαστικές μεθόδους και ελεγχόμενη τυχαιότητα, για την τοποθέτηση των πολύγονων εντός ενός κινούμενου ορίου-περιοχής τοποθέτησης, το οποίο μεταβάλλεται σταδιακά από το κάτω αριστερό έως το στο πάνω δεξί άκρο του περιβλήματος. Υλοποιήσαμε δύο γεωμετρικές παραλλαγές του κινούμενου ορίου: η μία χρησιμοποιούσε τετραγωνικό πλέγμα για την περιοχή τοποθέτησης, και η άλλη τμήματα κυκλικού τόξου.

Τα υποψήφια πολύγωνα κατατάχθηκαν σύμφωνα με συνάρτηση βαθμολόγησης, η οποία συνδύαζε πολλαπλά μεταδεδομένα, σταθμισμένα με την σημαντικότητά τους. Χρησιμοποιήθηκε εκθετική κατανομή για την επιλογή των πιο ευνοϊκών πολυγώνων σε κάθε βήμα. Οι πιθανές θέσεις τοποθέτησης εντός της κινούμενης περιοχής-ορίου, αξιολογήθηκαν με χρήση ευρετικών μεθόδων, για να επιλεχθεί η βέλτιστη επιλογή.

Η προσέγγιση μας, εμπνέεται από τις κλασικές bottom-left τεχνικές τοποθέτησης, μειώνοντας το πεδίο των λύσεων και αποφεύγοντας την ανάγκη για Non-Fit Polygon (NFP) υπολογισμούς, ελαχιστοποιώντας συνεπώς την πολυπλοκότητα υλοποίησης. Αν και αυτή η απλοποίηση μπορεί να οδηγήσει σε μείωση της ποιότητας συσκευασίας, προσφέρει έναν πρακτικό συμβιβασμό

μεταξύ υπολογιστικής αποδοτικότητας, ποιότητας λύσης και ευκολίας υλοποίησης.

Θεματική περιοχή: Υπολογιστική Γεωμετρία

Λέξεις κλειδιά: Βελτιστοποίηση, Γεωμετρική συσκευασία, Ευρετικές μέθοδοι, Στοχαστικές διαδικασίες, Κυρτό περίβλημα

ΕΠΙΒΛΕΠΩΝ
Ιωάννης Εμίρης, Καθηγητής ΕΚΠΑ

1 ΕΙΣΑΓΩΓΗ

Τα προβλήματα συσκευασίας είναι μια κατηγορία υπολογιστικά δύσκολων προβλημάτων, και συχνά ταξινομούνται ως NP-hard ή ακόμα και $\exists\mathbb{R}$-πλήρη [1], λόγω των γεωμετρικών περιορισμών τους, και της συνδυαστικής έκρηξης πιθανών διαμορφώσεων, αλλά έχουν σημαντικές πρακτικές εφαρμογές, όπως στην αποτελεσματική αξιοποίηση υλικών, στον κατασκευαστικό τομέα, και στη χωροταξική βελτιστοποίηση διατάξεων, μεταξύ άλλων.

1.1 Ορισμός Προβλήματος

Το πρόβλημα με το οποίο ασχοληθήκαμε, ορίζεται ως εξής: Δεδομένης μιας κυρτής περιοχής P στο επίπεδο και ενός συνόλου απλών πολυγώνων $Q1, \dots, Qn$, όπου κάθε πολύγωνο Qi συνοδεύεται από μία τιμή ci, ζητείται να υπολογιστεί μια εφικτή (μη επικαλυπτόμενη) τοποθέτηση ενός υποσυνόλου $S \subseteq \{1, \dots, n\}$ των πολυγώνων Qi εντός της περιοχής P, χωρίς περιστροφή, έτσι ώστε να μεγιστοποιείται η συνολική τιμή $\sum_{i \in S} ci$.

2 ΚΑΘΙΕΡΩΜΕΝΕΣ ΜΕΘΟΔΟΙ

Οι τυπικές προσεγγίσεις για τη συσκευασία πολυγώνων διακρίνονται σε ακριβείς μεθόδους, οι οποίες περιορίζονται σε προβλήματα πολύ μικρής κλίμακας λόγω του υψηλού υπολογιστικού τους κόστους, και σε προσεγγιστικές τεχνικές, οι οποίες χρησιμοποιούνται σε μεγαλύτερα προβλήματα και περιλαμβάνουν γεωμετρικές ευρετικές και μεταευρετικές τεχνικές (γενετικοί αλγόριθμοι, simulated annealing κ.ά.), οι οποίες προσφέρουν ποιοτικές λύσεις σε μικρότερη χρονική κλίμακα. Κεντρικά εργαλεία σε αυτές τις μεθόδους αποτελούν η ραστεροποίηση, όπου ο έλεγχος επικαλύψεων βασίζεται σε πλέγμα (grid) μέσω της ψηφιοποίησης των σχημάτων, και τα No-Fit Polygons (NFPs), τα οποία ορίζουν με γεωμετρική ακρίβεια τις επιτρεπτές θέσεις τοποθέτησης μεταξύ δύο πολυγώνων χωρίς επικάλυψη.

2.1 Σχετική Βιβλιογραφία

Η βιβλιογραφία για την τοποθέτηση ακανόνιστων σχημάτων προσφέρει μια ολοκληρωμένη χαρτογράφηση του πεδίου. Οι Bennell και Oliveira (2009) παρέχουν μια εκπαιδευτική ανασκόπηση των γεωμετρικών τεχνικών (NFPs, ραστεροποίηση) και των μεθόδων συσκευασίας [2]. Οι Leao et al. (2020) εστιάζουν στη μαθηματική μοντελοποίηση, παρουσιάζοντας πλήθος τεχνικών, συμπεριλαμβανομένων των μεθόδων μικτού ακέραιου και περιοριστικού προγραμματισμού [3]. Τέλος, οι Guo et al. (2022) παρουσιάζουν μια σύγχρονη επισκόπηση των αλγορίθμων βελτιστοποίησης και των προκλήσεων του πεδίου [4]. Συνολικά, οι παραπάνω ανασκοπήσεις παρουσιάζουν και χαρτογραφούν το πεδίο της δισδιάστατης συσκευασίας.

Παρά το γεγονός ότι οι υφιστάμενες μέθοδοι επιτυγχάνουν υψηλή πυκνότητα συσκευασίας, συχνά παρουσιάζουν αυξημένη υπολογιστική πολυπλοκότητα και είναι περίπλοκες στην υλοποίηση λόγω της ανάγκης για εξειδικευμένες μαθηματικές δομές (NFPs κ.α.).

2.2 Η Εξεταζόμενη Προσέγγιση στο Τοπίο των Λύσεων

Η προσέγγισή με την οποία πειραματιστήκαμε, χρησιμοποιεί στοχαστικές επιλογές, και αφορά την δημιουργία αρχικής λύσης, και όχι την βελτίωση

υπάρχουσας συσκευασίας. Αυτή η μέθοδος, που αποτελεί γεωμετρική ευρετική, αποφεύγει τον υπολογισμό NFPs ή την υποχρεωτική χρήση πλέγματος ραστεροποίησης / διακριτοποίησης, απλοποιώντας την υλοποίηση και βελτιώνοντας την απόδοση.

Χρησιμοποιώντας κινητό όριο τοποθέτησης, χωρίζει το δοχείο σε συσκευασμένη και προς συσκευασία περιοχές, επιτρέποντας πυκνότερες λύσεις από την απλή στοχαστική προσέγγιση, αλλά και μικρότερη υπολογιστική πολυπλοκότητα από πολλές bottom-left τεχνικές, καθώς το πεδίο ελέγχου για νέες πιθανές και έγκυρες θέσεις τοποθετήσεις περιορίζεται και δεν εξαρτάται άμεσα από κάθε ήδη τοποθετημένο πολύγωνο.

3 ΠΡΟΣΕΓΓΙΣΗ ΚΙΝΗΤΟΥ ΟΡΙΟΥ ΤΟΠΟΘΕΤΗΣΗΣ

Η προσέγγιση που παρουσιάζεται δέχεται ως είσοδο ένα κενό δοχείο και ένα σύνολο πολυγώνων προς τοποθέτηση. Ο αλγόριθμος βασίζεται σε μια μεταβαλλόμενη περιοχή - όριο τοποθέτησης, το οποίο μετακινείται προοδευτικά από το κάτω αριστερό προς το άνω δεξί όριο του δοχείου. Σε κάθε βήμα, τα πολύγωνα επιλέγονται με πιθανοτική διαδικασία σύμφωνα με σειρά προτεραιότητας που βασίζεται στα μεταδεδομένα και τα γεωμετρικά τους χαρακτηριστικά. Για κάθε επιλεγμένο πολύγωνο, εξετάζονται πολλαπλές τυχαίες θέσεις εντός της τρέχουσας περιοχής και επιλέγεται εκείνη που βελτιστοποιεί ευρετικές γεωμετρικές συναρτήσεις.

Για τον ορισμό της περιοχής τοποθέτησης εξετάζονται δύο εναλλακτικές:

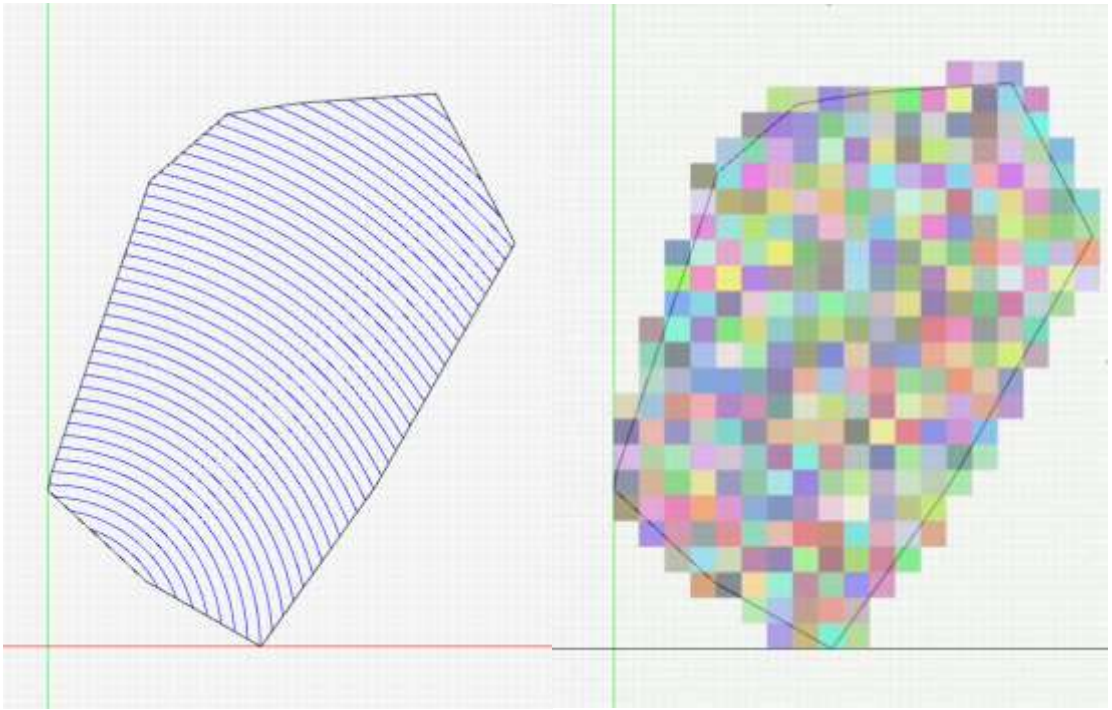
- Προσέγγιση τόξων – χρήση ομόκεντρων κυκλικών τόξων με κέντρο το κάτω αριστερό σημείο του δοχείου και σταδιακά αυξανόμενη ακτίνα
- Προσέγγιση πλέγματος – χρήση τετραγωνικών κυψελών πλεγματικής δομής

3.1 Μέθοδος Κυκλικών Τόξων

Η περιοχή τοποθέτησης των πολυγώνων ορίζεται ως δακτυλιοειδής ζώνη πέριξ κυκλικού τόξου, περιορισμένη εντός των ορίων του δοχείου. Το κέντρο του τόξου αντιστοιχεί στο κατώτερο αριστερό σημείο του περιβλήματος. Η ακτίνα του τόξου αυξάνεται σταδιακά σε κάθε βήμα, καλύπτοντας ομοιόμορφα όλη την περιοχή του δοχείου έως την άνω δεξιά γωνία.

3.2 Μέθοδος Τετραγώνων Πλεγματικής Δομής

Η περιοχή τοποθέτησης των πολυγώνων ορίζεται ως τετραγωνική κυψελίδα πλέγματος, με τις κυψελίδες να επιλέγονται διαδοχικά σε κάθε βήμα βάσει καθορισμένου μοτίβου πλήρωσης, ξεκινώντας από το κάτω αριστερό και προχωρώντας έως το άνω δεξί όριο του δοχείου. Κάθε κυψελίδα εξετάζεται για τοποθέτηση πολυγώνων με πολλαπλές επαναλήψεις, πριν η διαδικασία προχωρήσει στην επόμενη κυψελίδα.

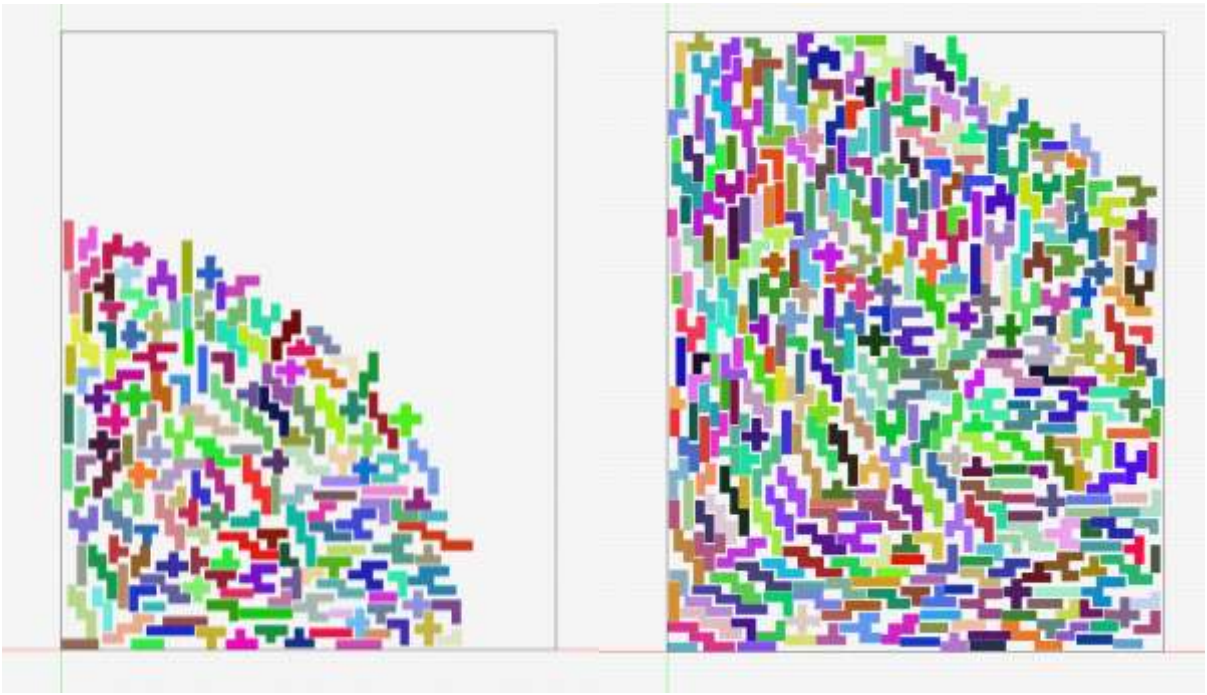


Εικόνα 1: Απεικόνιση κυκλικών τόξων τοποθέτησης και τετραγώνων πλεγματικής δομής

3.3 Σειρά Επιλογής Πολυγώνων και Ευρετική Καλύτερης Τοποθέτησης

Η σειρά ταξινόμησης και επιλογής των πολυγώνων καθορίζεται από μια συνάρτηση βαθμολόγησης (scoring function) βάσει μεταδεδομένων (όπως αξία και γεωμετρικά χαρακτηριστικά), με την τελική επιλογή να πραγματοποιείται στοχαστικά μέσω εκθετικής κατανομής ώστε να ευνοούνται τα ιδανικότερα σχήματα.

Για την επιλογή της θέσης τοποθέτησης, εντός της κινούμενης περιοχής, εφαρμόζονται ευρετικές μέθοδοι που στοχεύουν στην πυκνότερη διάταξη. Ενώ η απλούστερη προσέγγιση είναι η τυχαία επιλογή έγκυρης θέσης, η χρήση συναρτήσεων κόστους, (π.χ. ελαχιστοποίηση της απόστασης από το κάτω αριστερό άκρο του δοχείου) επιτρέπει την επιλογή ποιοτικότερων λύσεων.



Εικόνα 2: Στιγμιότυπα συσκευασίας μεθόδου κυκλικών τόξων

3.4 Αποδοτικός Έλεγχος Επικαλύψεων

Για την αποτελεσματική ανίχνευση επικαλύψεων, χρησιμοποιείται διφασική προσέγγιση. Κατά την Ευρεία Φάση (Broad Phase) χρησιμοποιούνται χωρικές δομές, με στόχο την ταχεία απόρριψή της πλειοψηφίας των μη συγκρουόμενων σχημάτων. Κατά τη Φάση Ακριβείας (Narrow Phase), εφαρμόζονται ακριβείς

γεωμετρικοί αλγόριθμοι, όπως GJK ή SAT σε κάθε πολύγωνο της πλέων σημαντικά μειωμένης λίστας των πιθανών επικαλυπτόμενων πολυγώνων.

Πιο συγκεκριμένα, κατά την ευρεία φάση πιθανές χωρικές δομές συμπεριλαμβάνουν:

- Quadtrees, R-trees, οι οποίες αξιοποιούν αποτελεσματικά τη μνήμη, αλλά έχουν αυξημένη χρονική πολυπλοκότητα αναζήτησης και ενημέρωσης ($O(\log n)$).
- Spatial Hash Grids (SHG), τα οποία προσφέρουν προσεγγιστικά σταθερό χρόνο αναζήτησης και ενημέρωσης ($O(1)$), αλλά με κόστος την υψηλότερη χωρική πολυπλοκότητα.

Εάν η χρήση μνήμης δεν αποτελεί περιορισμό, τα SHG προτιμώνται για την ανώτερη απόδοση σε αναζητήσεις και ενημερώσεις.

Τέτοιες δομές επιτρέπουν την εύρεση σχημάτων που βρίσκονται εντός μιας τοπικής περιοχής, πέριξ του υποψηφίου πολυγώνου. Αυτή η περιοχή μπορεί να είναι το AABB του πολυγώνου, ή τα επιμέρους τετράγωνα της διακριτοποιημένης (rasterized) μορφής του, αν στενότερη περιοχή είναι επιθυμητή.

Επιπλέον, μια απλή βελτιστοποίηση που προσφέρει το κινητό όριο με την μέθοδο των τόξων, είναι η αποφυγή ελέγχου των σχημάτων όπου η άνω δεξιά τους κορυφή είναι μικρότερη της της ελάχιστης ακτίνας της περιοχής τοποθέτησης. Αυτό μπορεί να επιτευχθεί χωρίς την αύξηση της χρονικής πολυπλοκότητας, χρησιμοποιώντας κουβάδες (buckets) για κάθε βήμα τοποθέτησης που συμπεριλαμβάνουν τα πολύγωνα που μελλοντικά δεν θα αποτελούν συγκρούσεις, αφαιρώντας τα από το SHG, και βελτιώνοντας τις χωρικές απαιτήσεις του αλγορίθμου.

Σε κάθε περίπτωση η επιλογή της στρατηγικής ελέγχου εξαρτάται από την επιθυμητή ισορροπία μεταξύ ταχύτητας αναζήτησης και ενημέρωσης, χρήσης μνήμης και απλότητας υλοποίησης.

3.5 Υπερπαράμετροι

Οι υπερπαράμετροι του αλγορίθμου καθορίζουν την συμπεριφορά του αλγορίθμου, όπως τη σχέση μεταξύ ταχύτητας και ποιότητας, καθώς, για παράδειγμα, η χρήση μικρότερου βήματος προώθησης του ορίου αυξάνει την πυκνότητα της συσκευασίας, αλλά με ανάλογη επιβάρυνση στον υπολογιστικό χρόνο.

Εικόνα 3: Μικρότερο βήμα περιοχής οδηγεί σε πυκνότερες συσκευασίες

3.6 Ψευδοκώδικας

```
Είσοδος: Δοχείο C, Σύνολο πολυγώνων P, Υπερπαράμετροι H, BT ∈{Τόξα, Πλέγμα}
Έξοδος: PackedPolygons ⊆ P

Αρχικοποίησε το όριο τοποθέτησης B με βάση το BT
Αρχικοποίησε PackedPolygons ← ∅


// Προεπεξεργασία
ΓΙΑ κάθε πολύγωνο p ∈ P:
    Υπολόγισε μεταδεδομένα (τιμή, εμβαδόν, squareness)
    Υπολόγισε αρχικό score(p)
Ταξινόμησε τα πολύγωνα κατά φθίνουσα σειρά score
```

```
ΟΣΟ το όριο B δεν έχει διατρέξει πλήρως το δοχείο ΚΑΙ P ≠ ∅:

    // Βήμα 1: Επιλογή υποψήφιων πολυγώνων
    CandidatePolygons ← Τυχαίο υποσύνολο του P με έμφαση στα
                        υψηλής βαθμολογίας πολύγωνα

    // Βήμα 2: Προσπάθεια τοποθέτησης
    ΓΙΑ κάθε πολύγωνο p ∈ CandidatePolygons:
        Εκτέλεσε H.num_trials τυχαίες δοκιμές τοποθέτησης εντός του B
        Αξιολόγησε τις έγκυρες θέσεις με ευρετική συνάρτηση κόστους
        Επίλεξε την καλύτερη έγκυρη θέση (αν υπάρχει)
        ΑΝ έγκυρη θέση έχει βρεθεί:
            Τοποθέτησε το p στο C και πρόσθεσέ το στο PackedPolygons
            Αφαίρεσε το p από το P
        ΑΛΛΙΩΣ:
            Αναπροσάρμοσε το score(p) ώστε να μειωθεί η προτεραιότητά του

    // Βήμα 3: Προώθηση ορίου
    Μετάφερε το όριο B κατά H.boundary_step

ΕΠΕΣΤΡΕΨΕ PackedPolygons
```

3.7 Υπολογιστική Πολυπλοκότητα

Για τον υπολογισμό της χρονικής πολυπλοκότητας, έστω: n πλήθος πολυγώνων, t δοκιμές ανά πολύγωνο, m βήματα ορίου, k πολύγωνα ανά βήμα ($k \ll n$) και c το κόστος ελέγχου επικάλυψης. Η πολυπλοκότητα αναλύεται ως εξής:

- **Υπολογισμός μεταδεδομένων και ταξινόμηση:** $O(n)$ και $O(n \log n)$ αντίστοιχα
- **Κύριος βρόχος τοποθέτησης:** Σε κάθε βήμα $i = 1 \dots m$, εξετάζονται k πολύγωνα, με t δοκιμές ανά πολύγωνο και κόστος $O(c)$ ανά δοκιμή. Το συνολικό κόστος ανά βήμα είναι: $O(k \cdot t \cdot c)$. Συνολικά: $O(m \cdot k \cdot t \cdot c)$.

Συνεπώς, συνολικά, η πολυπλοκότητα του αλγορίθμου εκφράζεται ως:

$$O(n \log n + m \cdot k \cdot t \cdot c).$$

Έστω $A = m \cdot k \cdot t$, τότε η πολυπλοκότητα εκφράζεται ως:

$$O(n \log n + A \cdot c).$$

Η στρατηγική ελέγχου επικαλύψεων καθορίζει το υπολογιστικό κόστος ανά τοποθέτηση και, κατ' επέκταση, τη συνολική πολυπλοκότητα. Με τη χρήση

SHGs, ο έλεγχος πραγματοποιείται σε σχεδόν σταθερό χρόνο $c = O(1)$. Συνεπώς στην πιο αποδοτική του μορφή ο αλγόριθμος έχει πολυπλοκότητα:

$$O(n \log n + A)$$

Συνεπώς, η χρονική πολυπλοκότητα του αλγορίθμου συνοψίζεται σε δύο συνιστώσες: τον χρόνο ταξινόμησης των πολυγώνων $O(n \log n)$ και το κόστος τοποθέτησης $O(A)$. Σε περιπτώσεις όπου η αρχική ταξινόμηση δεν απαιτείται , π.χ. όταν όλα τα πολύγωνα έχουν ίση αξία ή παρόμοια χαρακτηριστικά, η πολυπλοκότητα απλοποιείται σε $O(A)$, δηλαδή είναι γραμμική ως προς το εμβαδό και την πυκνότητα συσκευασίας.

Ο διαχωρισμός του δοχείου σε γεμάτη, προς τοποθέτηση και κενή περιοχή, σε συνδυασμό με την στοχαστική τοποθέτηση των σχημάτων, περιορίζει την εξάρτηση μεταξύ πολυγώνων. Αυτό διαφοροποιεί τη μέθοδο από κλασικές τεχνικές συσκευασίας, όπως το bottom-left με NFPs, όπου κάθε νέο πολύγωνο πρέπει να ελεγχθεί έναντι όλων των ήδη τοποθετημένων, με αποτέλεσμα τετραγωνικό υπολογιστικό κόστος.

3.8 Βελτιώσεις και Επεκτάσεις

Υπάρχουν διάφορες πιθανές βελτιώσεις που μπορούν να εξεταστούν, με στόχο την περαιτέρω αύξηση της αποδοτικότητας, ποιότητας, και ευελιξίας της συσκευασίας.

Αρχικά, ο αλγόριθμος είναι ευπροσάρμοστος σε διάφορες παραλλαγές του προβλήματος λόγω της απλότητας και της γεωμετρικής του φύσης. Η αρχή του κινούμενου ορίου μπορεί να επεκταθεί για κοίλα ή πιο περίπλοκα δοχεία και προβλήματα υψηλότερων διαστάσεων (π.χ. 3D συσκευασία). Επίσης, μπορεί να προσαρμοστεί για σύνθετα γεωμετρικά σχήματα (π.χ. πολύγωνα με οπές ή καμπύλα όρια), αξιοποιώντας κατάλληλες συναρτήσεις ανίχνευσης συγκρούσεων και ευρετικές τοποθέτησης.

Επιπλέον, μια πιθανή κατεύθυνση βελτίωσης αφορά τη βελτιστοποίηση της σειράς τοποθέτησης των πολυγώνων. Λόγω της στοχαστικής φύσης του αλγορίθμου, η ίδια ακολουθία μπορεί να αποδίδει διαφορετικά αποτελέσματα σε κάθε εκτέλεση, γεγονός που καθιστά δύσκολη την βελτιστοποίησή της σειράς

τοποθέτησης. Μια πιο αποτελεσματική προσέγγιση θα μπορούσε να βασιστεί σε τοπικές τροποποιήσεις και μερική συνέχιση από ενδιάμεσες καταστάσεις του αλγορίθμου, αξιοποιώντας μεταευρετικές τεχνικές όπως γενετικούς αλγορίθμους ή προσομοιωμένη ανόπτηση, επιτρέποντας τη σταδιακή βελτίωση των πιο υποσχόμενων λύσεων και τη μείωση της επίδρασης της τυχαιότητας.

Εναλλακτικά, μια πιθανή επέκταση θα μπορούσε να είναι η ενσωμάτωση τοπικών αλγορίθμων ακριβείας, όπως η μέθοδος bottom-left με NFP, εντός των επιμέρους περιοχών τοποθέτησης που ορίζονται από το κινούμενο όριο. Με αυτόν τον τρόπο, αξιοποιείται το πλεονέκτημα του κινούμενου ορίου τοποθέτησης, καθώς η υπολογιστικά απαιτητική διαδικασία περιορίζεται στην ενεργή περιοχή, ενώ η ήδη συσκευασμένη ζώνη εξαιρείται από τους ελέγχους. Η τοπική εφαρμογή NFP μπορεί να μειώσει τους ελέγχους σύγκρουσης, διατηρώντας ικανοποιητική πυκνότητα συσκευασίας.

4 ΑΞΙΟΛΟΓΗΣΗ ΠΡΟΣΕΓΓΙΣΗΣ

Παρακάτω εξετάζεται η απλοϊκή εκδοχή του αλγορίθμου, χωρίς τη χρήση πρόσθετων βελτιστοποιήσεων ή μεταευρετικών τεχνικών. Τα αποτελέσματα δείχνουν ότι η προσέγγιση μας παρουσιάζει χαμηλότερη ποιότητα συσκευασίας σε σύγκριση με πιο εξελιγμένες, ντετερμινιστικές μεθόδους τοποθέτησης, μερικές από τις οποίες αξιοποιούν μεταευρετικές τεχνικές βελτιστοποίησης. Ωστόσο, η υλοποίηση του αλγορίθμου παρέχει εύκολη παραμετροποίηση της σχέσης χρόνου εκτέλεσης προς ποιότητας συσκευασίας, είναι απλή στην εφαρμογή και διατηρεί ικανοποιητικό χρόνο εκτέλεσης.

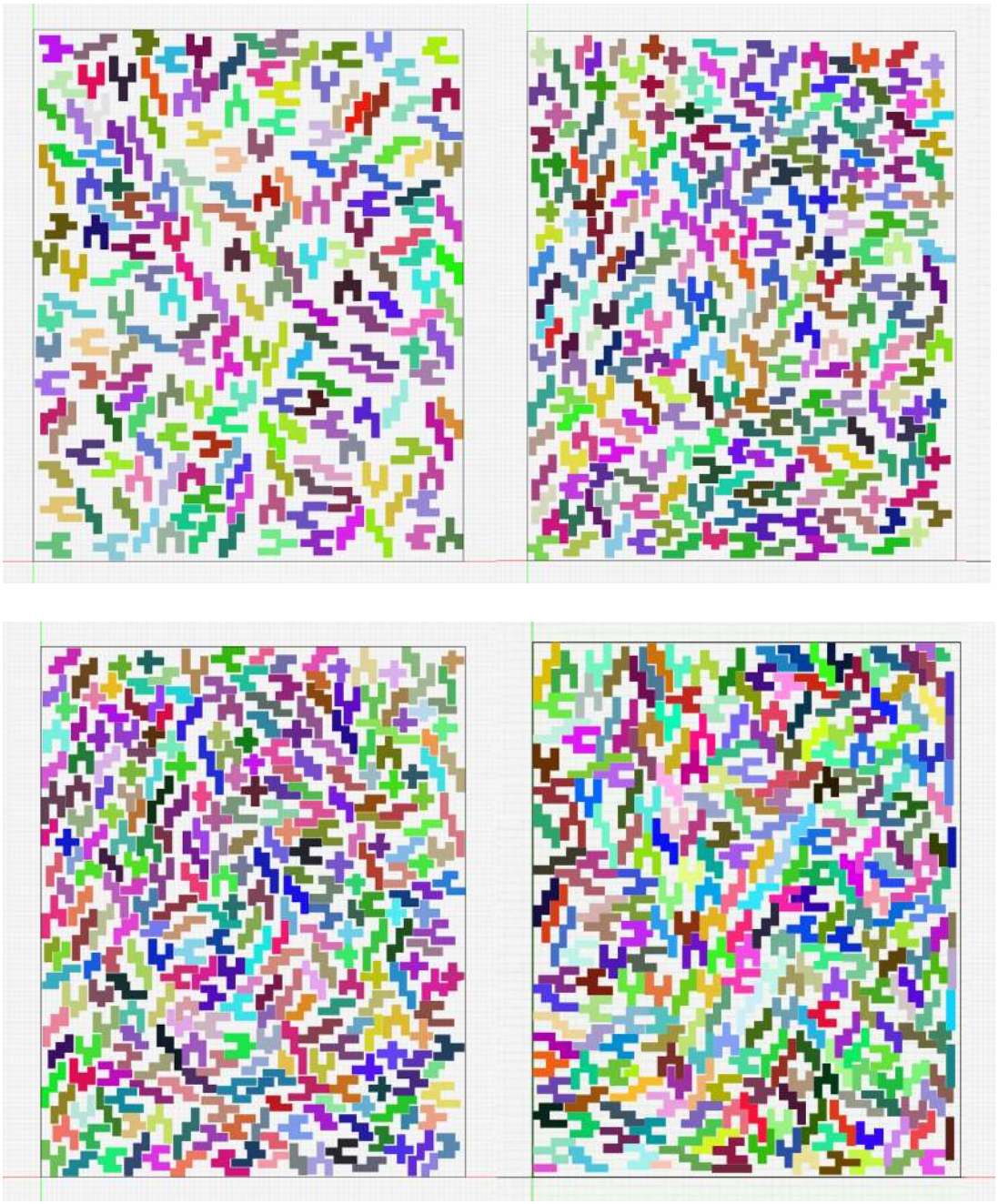
4.1 Σύγκριση Αλγορίθμων

Για την σύγκριση και αξιολόγηση του αλγορίθμου, εξετάστηκαν τέσσερις προσεγγίσεις: (α) αφελής, πλήρως τυχαία τοποθέτηση, (β) στοχαστική συσκευασία με κινούμενο όριο και μεγάλο βήμα, (γ) στοχαστική συσκευασία με κινούμενο όριο και μικρό βήμα, και (δ) τοποθέτηση bottom-left με NFP. Παρατηρείται ότι, ακόμη και με μεγάλο χρόνο εκτέλεσης, η πλήρως τυχαία

τοποθέτηση περιορίζεται από τα κενά που δημιουργούνται μεταξύ σχημάτων παρόμοιου μεγέθους, τα οποία δεν μπορούν να καλυφθούν από μικρότερα πολύγωνα. Η προσέγγιση του κινούμενου ορίου επιτυγχάνει καλή ισορροπία μεταξύ ταχύτητας και ποιότητας συσκευασίας, βελτιώνοντας αισθητά τα αποτελέσματα σε σχέση με την αφελή τυχαία τοποθέτηση, ενώ το μέγεθος του βήματος καθορίζει τον συμβιβασμό μεταξύ χρονικής απόδοσης και πυκνότητας. Τέλος, η μέθοδος bottom-left με NFP οδηγεί σε εμφανώς πυκνότερη συσκευασία, με αντίτιμο την αυξημένη πολυπλοκότητα υλοποίησης και τον μεγαλύτερο χρόνο εκτέλεσης.

Πίνακας 1: Σύγκριση απόδοσης αλγορίθμων

Αλγόριθμος	**Αριθμός Σχημάτων**	**Σχετικό Ποσοστό Σχημάτων**	**Ποσοστό συσκευασίας (Εμβαδό)**	**Χρόνος εκτέλεσης (ενδεικτικός)**
Αφελής, τελείως στοχαστικός	181	100%	48%	8,0 sec.
Κινητό όριο τοποθέτησης (μεγάλο βήμα)	255	141%	56%	4,5 sec.
Κινητό όριο τοποθέτησης (μικρό βήμα)	294	162%	63%	15,2 sec.
Bottom-left fill με NFP και GA=1 γενιά (SVG Nest)	269	149%	66%	115,1 sec.



Εικόνα 4: Αποτελέσματα, από αριστερά: (α) Στοχαστικός, (β) Κινητό όριο, μεγάλο βήμα, (γ) Κινητό όριο, μικρό βήμα, (δ) Bottom-left με NFP

Οι χρόνοι εκτέλεσης του Πίνακα 1 δεν είναι άμεσα συγκρίσιμοι, καθώς οι υλοποιήσεις βασίζονται σε διαφορετικά περιβάλλοντα και τεχνολογίες. Παρέχονται ως ένδειξη της τάξης μεγέθους και όχι ως απόλυτο μέτρο απόδοσης.

4.2 Αποτελέσματα Διαγωνισμού

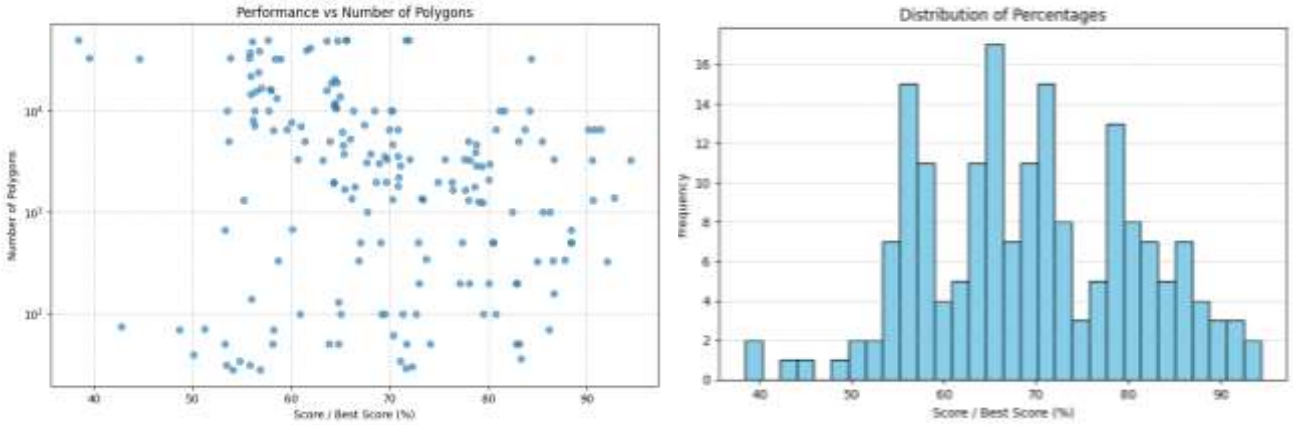
Ο αλγόριθμος αξιολογήθηκε στο πλαίσιο του διαγωνισμού, παρουσιάζοντας μέση επίδοση περίπου στο 70% των καλύτερων συμμετεχόντων αποτελεσμάτων (Best score). Παρά την χαμηλή του απόδοση, ο αλγόριθμος επέδειξε σταθερή συμπεριφορά σε διαφορετικά σύνολα δεδομένων, επιτρέποντας την εξαγωγή χρήσιμων συμπερασμάτων για τα όρια και τις δυνατότητές του.

Αναλυτικότερα, το ανώτερο 10% των αποτελεσμάτων προσέγγισε το 90% της ποιότητας συσκευασίας των βέλτιστων επιδόσεων, ενώ αντίστοιχα το κατώτερο 10% κυμάνθηκε στο 50% του βέλτιστου σκορ, υποδεικνύοντας σημαντική διακύμανση στα αποτελέσματα. Πιο συγκεκριμένα, ο μέσος λόγος απόδοσης ανήλθε σε 69.38%, με διάμεσο 69.13%, επιβεβαιώνοντας τη συνοχή των αποτελεσμάτων γύρω από τη μέση τιμή. Η τυπική απόκλιση (11.56%) και το ενδοτεταρτημοριακό εύρος (18.19%) καταδεικνύουν αξιοσημείωτη διασπορά στις επιδόσεις μεταξύ διαφορετικών περιπτώσεων, στοιχείο που συμφωνεί με τη στοχαστική φύση του αλγορίθμου.

Η παρατηρούμενη διασπορά υποδηλώνει ότι η απόδοση του αλγορίθμου ίσως θα μπορούσε να βελτιωθεί μέσω επαναλαμβανόμενων εκτελέσεων με διαφορετικές τυχαίες διαδικασίες ή μέσω ενσωμάτωσης μεταευρετικών τεχνικών, οι οποίες θα επέτρεπαν την εκμετάλλευση των πιο υποσχόμενων λύσεων και τη σταδιακή βελτίωση της ποιότητας συσκευασίας.

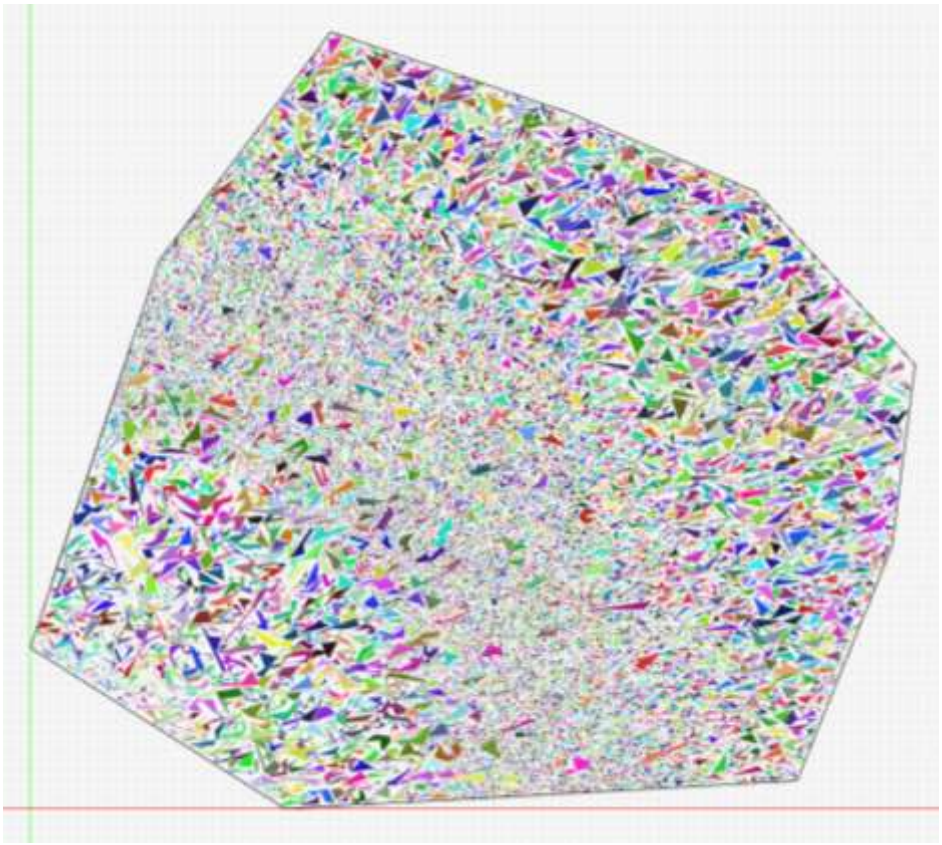
Όλες οι εκτελέσεις πραγματοποιήθηκαν σε παράλληλα νήματα. Οι περισσότερες περιπτώσεις ολοκληρώθηκαν σε λίγα δευτερόλεπτα ή λεπτά, ενώ τα πολύ απαιτητικά σύνολα δεδομένων χρειάστηκαν πάνω από μία ώρα, ανάλογα με το μέγεθος των πολυγώνων και το εμβαδό του δοχείου συσκευασίας.

Στο Σχήμα 1 παρουσιάζεται η σχέση του λόγου επίδοσης (Score / Best Score) με το πλήθος των πολυγώνων κάθε προβλήματος, όπου παρατηρείται ελαφρά αρνητική συσχέτιση ($r = -0.36$, $p < 10^{-6}$). Το ιστόγραμμα του απεικονίζει την κατανομή της σχετικής απόδοσης, η οποία συγκεντρώνεται γύρω από το 70% του βέλτιστου, με περιορισμένο αριθμό εξαιρετικά χαμηλών ή υψηλών τιμών.



Σχήμα 1: Σχέση επίδοσης προς αριθμό πολυγώνων και ιστόγραμμα κατανομής

Συμπερασματικά, παρά τη μειωμένη ποιότητα συσκευασίας στον διαγωνισμό, ο αλγόριθμος επιδεικνύει ικανοποιητική λειτουργικότητα και σταθερότητα, σε διάφορα σενάρια τοποθέτησης. Επιπλέον, η διασπορά στην απόδοση συσκευασίας ίσως υποδηλώνει πως υπάρχουν περιθώρια βελτίωσης, για την αύξηση της απόδοσης του σκορ συσκευασίας.



Εικόνα 5: Παράδειγμα συσκευασίας συνόλου πολυγώνων με κοιλότητες

5 ΣΥΜΠΕΡΑΣΜΑΤΑ

Συμπερασματικά, η προσέγγιση χρησιμοποιεί κινητό όριο τοποθέτησης για τη μείωση της πολυπλοκότητας επιλογής θέσης, και στοχαστική τοποθέτηση για τη διατήρηση της απλότητας του αλγορίθμου. Παρουσιάζει συμβιβασμούς μεταξύ ποιότητας συσκευασίας, χρόνου εκτέλεσης και ευκολίας υλοποίησης. Η ποιότητα συσκευασίας είναι αισθητά χαμηλότερη από πιο εξελιγμένες, ντετερμινιστικές μεθόδους τοποθέτησης, και η απόδοση επηρεάζεται από την επιλογή κατάλληλων υπερπαραμέτρων. Ο αλγόριθμος παραμένει παραμετροποιήσιμος, απλός στην υλοποίηση, ευέλικτος σε διαφορετικά σενάρια (υψηλότερες διαστάσεις, κοίλα δοχεία) και προσφέρει αποδεκτούς χρόνους εκτέλεσης.

Οι πιθανές του χρήσεις περιορίζονται σε περιπτώσεις όπου η ταχύτητα εκτέλεσης ή η απλότητα υπερτερούν της μέγιστης πυκνότητας συσκευασίας, όπως σε περιβάλλοντα με χαμηλή επεξεργαστική ισχύ και περιορισμένες βιβλιοθήκες υπολογιστικής γεωμετρίας (π.χ. ενσωματωμένα συστήματα), εφαρμογές με απαιτήσεις γρήγορης απόκρισης και εκπαιδευτικούς σκοπούς.

Πιθανές μελλοντικές βελτιώσεις περιλαμβάνουν την ενσωμάτωση μεταευρετικών τεχνικών, καθώς και τοπική χρήση ντετερμινιστικών αλγορίθμων ακριβείας σε κάθε θέση του κινούμενου ορίου, αξιοποιώντας την χαμηλότερη πολυπλοκότητα του.

ΑΝΑΦΟΡΕΣ

[1] M. Abrahamsen, T. Miltzow and N. Seiferth, "Framework for ER-Completeness of Two-Dimensional Packing Problems," 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), Durham, NC, USA, 2020, pp. 1014-1021, doi: 10.1109/FOCS46700.2020.00098.

[2] J. A. Bennell and J. F. Oliveira, 'A tutorial in irregular shape packing problems', Journal of the Operational Research Society, vol. 60, no. sup1, pp. S93–S105, 2009.

[3] A. A. S. Leao, F. M. B. Toledo, J. F. Oliveira, M. A. Carravilla, and R. Alvarez-Valdés, 'Irregular packing problems: A review of mathematical models', European Journal of Operational Research, vol. 282, no. 3, pp. 803–822, 2020.

[4] B. Guo et al., 'Two-dimensional irregular packing problems: A review', Frontiers in Mechanical Engineering, vol. 8, p. 966691, 2022.

Ανάπτυξη Εικονικών Συσκευών σε Λειτουργικά Συστήματα Windows

Μάριος-Τριαντάφυλλος Καρναβάς

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία αναλύει τη διαδικασία ανάπτυξης εικονικών συσκευών σε λειτουργικό σύστημα Windows, στις εκδοχές Windows 7, 8 και 10. Συγκεκριμένα, για την εργασία αυτή αναπτύχθηκε μία εικονική κάμερα μεταβλητής ανάλυσης, της οποίας τα δεδομένα εξόδου μπορούν να μεταβληθούν από άλλα προγράμματα. Υλοποιήθηκαν, επίσης, ενδεικτικά παραδείγματα προγραμμάτων που μεταβάλλουν την ανάλυση και τα καρέ που προβάλει. Με το πέρας της διαδικασίας ανάπτυξης της κάμερας, πραγματοποιήθηκε έλεγχος απόδοσής της. Πραγματοποιήθηκε μελέτη και στην ανάπτυξη εικονικών μικροφώνων μέσω ανάλυσης του κώδικα, των λειτουργιών και των επιδόσεων ορισμένων δημοσίως διαθέσιμων εικονικών συσκευών.

ΕΠΙΒΛΕΠΩΝ
Θεοχάρης Θεοχάρης, Καθηγητής ΕΚΠΑ

1 ΕΙΣΑΓΩΓΗ

Η παρούσα εργασία πραγματεύεται την ανάπτυξη εικονικών συσκευών σε λειτουργικά συστήματα της οικογένειας Windows (Windows 7, 8 και 10), με έμφαση στη δημιουργία μίας εικονικής κάμερας. Οι εικονικές συσκευές αποτελούν λογισμικές οντότητες που προσομοιώνουν τη λειτουργία φυσικών συσκευών υλικού, επιτρέποντας στο λειτουργικό σύστημα και στις εφαρμογές να αλληλεπιδρούν μαζί τους σαν να ήταν πραγματικές.

Η ανάγκη για εικονικές συσκευές έχει αυξηθεί σημαντικά τα τελευταία χρόνια, ιδίως λόγω της εξάπλωσης εφαρμογών τηλεδιάσκεψης, επεξεργασίας πολυμέσων και ανάπτυξης αλγορίθμων επεξεργασίας εικόνας και ήχου σε πραγματικό χρόνο. Η δυνατότητα ελέγχου και τροποποίησης των δεδομένων που «παράγει» μία συσκευή εισόδου χωρίς να απαιτείται φυσική παρέμβαση προσφέρει σημαντικά πλεονεκτήματα τόσο σε ερευνητικό όσο και σε πρακτικό επίπεδο.

Κύριος στόχος της εργασίας ήταν να διερευνηθεί κατά πόσο είναι εφικτή η ανάπτυξη μίας λειτουργικής και επεκτάσιμης εικονικής συσκευής μέσα σε εύλογο χρονικό διάστημα, χρησιμοποιώντας τα εργαλεία που προσφέρει το λειτουργικό σύστημα των Windows. Επιπλέον, διερευνήθηκαν οι διαφορές μεταξύ των τεχνολογιών που χρησιμοποιούνται για την ανάπτυξη των εικονικών συσκευών, καθώς και οι περιπτώσεις στις οποίες είναι προτιμότερη η επιλογή της καθεμίας.

2 ΥΠΟΒΑΘΡΟ

2.1 Ορισμοί και Μέθοδοι Αναπτυξης

Μία εικονική συσκευή ορίζεται ως συνιστώσα λογισμικού που προσομοιώνει τη λειτουργία μίας συσκευής υλισμικού [1]. Στην παρούσα εργασία εξετάζονται κυρίως εικονικές συσκευές εισόδου, δηλαδή προγράμματα που μιμούνται την εισαγωγή δεδομένων στο υπολογιστικό σύστημα [2].

Η ανάπτυξη εικονικών συσκευών μπορεί να γίνει με δύο βασικούς τρόπους:

- Μέσω ανάπτυξης προγράμματος οδήγησης συσκευής (device driver), το οποίο λειτουργεί συχνά σε επίπεδο πυρήνα (kernel-mode).
- Μέσω αξιοποίησης βιβλιοθηκών και εργαλείων υψηλότερου επιπέδου που παρέχει το λειτουργικό σύστημα.

Η πρώτη προσέγγιση προσφέρει αυξημένο έλεγχο και ενδεχομένως καλύτερες επιδόσεις, αλλά ενέχει αυξημένο κίνδυνο για τη σταθερότητα του συστήματος.

Αντίθετα, η δεύτερη προσέγγιση προσφέρει μεγαλύτερη ασφάλεια, ταχύτερη ανάπτυξη και ευκολότερη συντήρηση.

Στο πλαίσιο της παρούσας εργασίας επιλέχθηκε η χρήση της βιβλιοθήκης DirectShow, η οποία παρέχει αρθρωτή αρχιτεκτονική για τη διαχείριση πολυμέσων στα Windows, καθώς και τη δυνατότητα ορισμού εικονικών συσκευών εισόδου βίντεο.

2.2 Η Βιβλιοθήκη DirectShow

Η DirectShow βασίζεται στην έννοια των φίλτρων (filters) [3], τα οποία οργανώνονται σε γράφους φίλτρων (filter graphs). Κάθε φίλτρο επιτελεί συγκεκριμένο ρόλο:

- Φίλτρα Πηγής (Source Filters): Εισάγουν δεδομένα στο σύστημα του γράφου φίλτρων.
- Φίλτρα Μετασχηματισμού: Επεξεργάζονται δεδομένα που προέρχονται από φίλτρα πηγής ή άλλα φίλτρα μετασχηματισμού και τα προωθούν στο επόμενο φίλτρο του γράφου.
- Φίλτρα Απεικόνισης: Παρουσιάζουν τα τελικά αποτελέσματα στον χρήστη, χρησιμοποιώντας συνήθως διάφορες περιφερειακές συσκευές (οθόνη, ηχεία). Δεν προωθούν δεδομένα σε άλλα φίλτρα του γράφου.

Η επικοινωνία μεταξύ φίλτρων πραγματοποιείται μέσω καρφίων (pins), τα οποία ορίζουν τα σημεία εισόδου και εξόδου δεδομένων σε κάθε φίλτρο, καθώς και τους τύπους των δεδομένων που δέχεται και προωθεί [4]. Κάθε φίλτρο μπορεί να προσφέρει πολλαπλά καρφία εισόδου και εξόδου. Τα φίλτρα πηγής δεν περιέχουν καρφία εισόδου, ενώ τα φίλτρα απεικόνισης δεν έχουν καρφία εξόδου.

3 ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΙΚΟΝΙΚΗΣ ΚΑΜΕΡΑΣ

Η επιλογή της DirectShow κρίθηκε κατάλληλη, καθώς επιτρέπει τη δημιουργία φίλτρου πηγής που μπορεί να αναγνωριστεί από το σύστημα ως συσκευή

εισόδου βίντεο. Μέσω αυτής της δυνατότητας, το πρόβλημα της δημιουργίας μίας εικονικής κάμερας ανάγεται στο πρόβλημα δημιουργίας ενός φίλτρου πηγής, του οποίου τα δεδομένα παράγουν δυναμικά κάποια σειρά από καρέ. Ταυτόχρονα, επιδιώχθηκε το φίλτρο να εύκολα προσβάσιμο από προγραμματιστές. Για τους παραπάνω λόγους, το φίλτρο πηγής, πριν παράγει ένα καρέ, ελέγχει αρχικά αν ένας συγκεκριμένος χώρος κοινής μνήμης είναι ανοιχτός στο λειτουργικό σύστημα του υπολογιστή:

- Αν δεν είναι, τότε το φίλτρο παράγει απλά ένα ενιαίο, μονόχρωμο καρέ.
- Αν ο χώρος είναι ανοιχτός, το φίλτρο αντιγράφει και προωθεί τα δεδομένα που έχουν αποθηκευτεί στον χώρο κοινής μνήμης ως ένα συνεχές καρέ.

Η χρήση κοινής μνήμης (shared memory) επιτρέπει σε εξωτερικά προγράμματα να τροποποιούν το περιεχόμενο που προβάλλει η εικονική κάμερα. Με αυτόν τον τρόπο επιτυγχάνεται ο διαχωρισμός της βασικής λειτουργίας της κάμερας από την επεξεργασία των δεδομένων. Προβλήματα συγχρονισμού, όπως η ταυτόχρονη ανάγνωση και εγγραφή στην κοινή μνήμη, επιλύονται με τη χρήση ενός συγκεκριμένου mutex. Το mutex κλειδώνει την περιοχή της κοινής μνήμης όσο την χρησιμοποιεί κάποιο πρόγραμμα, υποχρεώνοντας τα υπόλοιπα προγράμματα να περιμένουν και διασφαλίζοντας ότι την προσπελαύνει το πολύ ένα πρόγραμμα κάθε χρονική στιγμή.

Η εικονική κάμερα υποστηρίζει επίσης την παραγωγή καρέ σε διαφορετικές αναλύσεις, χρησιμοποιώντας έναν διαφορετικό χώρο κοινής μνήμης. Κάθε στιγμιότυπο του φίλτρου, πριν δημιουργηθεί, ελέγχει αν αυτή η περιοχή έχει ανοιχτεί από κάποιο άλλο πρόγραμμα και, αν είναι, χρησιμοποιεί την ανάλυση που έχει γραφτεί σε εκείνη. Διαφορετικά, χρησιμοποιεί τη μέγιστη ανάλυση που υποστηρίζεται.

Η μέγιστη ανάλυση που υποστηρίζεται ορίστηκε στα 1280x960 εικονοστοιχεία, ενώ κάθε καρέ αποθηκεύεται ως ακολουθία τριάδων byte (RGB). Θεωρήθηκε ευθύνη των προγραμμάτων που παράγουν τα καρέ, δηλαδή τα καταγράφουν στην κοινή μνήμη, να συμμορφώνονται με τους παραπάνω περιορισμούς.

Η εγκατάσταση της εικονικής κάμερας διεκπεραιώνεται μέσω της δυναμικής βιβλιοθήκης που ορίζει τα χαρακτηριστικά και την λειτουργία της. Η βιβλιοθήκη αυτή οφείλει να προβάλλει ορισμένες συναρτήσεις, μεταξύ των οποίων οι DllRegisterServer και DllUnregisterServer, που ορίζουν την διαδικασία εγκατάστασης και απεγκατάστασης της εικονικής κάμερας. Για την τελική εγκατάσταση της εικονικής κάμερας, αρκεί να εκτελεστεί η εντολή:

regsvr32 <PathToDll>\VirtualCameraBase.dll

Η εντολή αυτή καλεί την συνάρτηση DllRegisterServer ώστε να καταγράψει την εικονική κάμερα στην λίστα των συσκευών εισόδου βίντεο του λειτουργικού συστήματος. Είναι αναγκαίο να χρησιμοποιηθεί, καθώς στο περιβάλλον που εκτελείται προσφέρονται ορισμένες συναρτήσεις σημαντικές για την λειτουργία της DllRegisterServer. Μετά από την εκτέλεσή της, το φίλτρο αναγνωρίζεται κανονικά ως συσκευή εισόδου βίντεο. Οι εφαρμογές που χρησιμοποιούν πολυμέσα μπορούν έπειτα να έχουν πρόσβαση σε στιγμιότυπά του και, κατ' επέκταση, να εμφανίσουν τα δεδομένα που παράγει, μέσω ειδικών συναρτήσεων και μεθόδων κατασκευής που ορίστηκαν στη βιβλιοθήκη. Η απεγκατάσταση της κάμερας γίνεται με παρόμοιο τρόπο, προστίθεται απλώς η παράμετρος -u στην εκτέλεση της παραπάνω εντολής.

4 ΔΥΝΑΜΙΚΗ ΡΥΘΜΙΣΗ ΑΝΑΛΥΣΗΣ

Ως παράδειγμα υλοποίησης, αναπτύχθηκε βοηθητικό πρόγραμμα που επιτρέπει τη ρύθμιση της ανάλυσης της εικονικής κάμερας. Το πρόγραμμα αναζητά στον υπολογιστή του χρήστη συσκευές εισόδου βίντεο (ουσιαστικά κάμερες), και δίνει τη δυνατότητα επιλογής μίας από τις υποστηριζόμενες αναλύσεις τους. Η πληροφορία αυτή αποθηκεύεται σε κοινή μνήμη, επιτρέποντας στην εικονική κάμερα να προσαρμόζει δυναμικά τον τύπο πολυμέσου που δηλώνει στις εφαρμογές.

Αν και η εικονική κάμερα δεν υποστηρίζει την αλλαγή ανάλυσης όσο προβάλλει δεδομένα, η προσέγγιση αυτή προσφέρει ευελιξία χωρίς να απαιτείται επανεγκατάσταση ή επανακαταχώρηση της συσκευής.

5 ΕΠΕΞΕΡΓΑΣΙΑ ΚΑΡΕ

5.1 Υλοποίηση Προγράμματος Επεξεργασίας Καρέ

Η αρχιτεκτονική που χρησιμοποιήθηκε για τη δημιουργία της εικονικής κάμερας επιτρέπει την εφαρμογή αλγορίθμων γραφικών σε πραγματικό χρόνο, με δυνατότητα χρήσης του αποτελέσματος σε εφαρμογές ζωντανής μετάδοσης εικόνας.

Ως παράδειγμα υλοποίησης αναπτύχθηκε πρόγραμμα που λαμβάνει καρέ από πραγματική κάμερα, εφαρμόζει προαιρετικά μετασχηματισμούς και τα διοχετεύει στην εικονική κάμερα. Το πρόγραμμα αρχικά ζητά από τον χρήστη να επιλέξει μία συσκευή εισόδου βίντεο που είναι συνδεδεμένη στο σύστημά του. Εφόσον επιλεγεί μία συμβατή κάμερα, το φίλτρο που την αντιπροσωπεύει στο σύστημα DirectShow συνδέεται με ένα φίλτρο το οποίο καταγράφει τα καρέ που του παρέχονται στην κοινή μνήμη. Το φίλτρο έχει επιπλέον την δυνατότητα να εφαρμόσει προαιρετικά έναν μετασχηματισμό σε κάθε καρέ, πριν το καταγράψει στην μνήμη, μεταβάλλοντας έτσι το τελικό αποτέλεσμα της ζωντανής ροής της εικονικής κάμερας. Παρακάτω προβάλλονται οι μετασχηματισμοί που υλοποιήθηκαν.

5.2 Μετασχηματισμοί Καρέ

5.2.1 Καθρεπτισμός εικόνας

Απλή αντιστροφή του καρέ ως προς τον κατακόρυφό του άξονα.

(a) Στιγμιότυπο της εικονικής κάμερας, χωρίς μετασχηματισμό.

(b) Στιγμιότυπο της εικονικής κάμερας, μετά τον μετασχηματισμό.

Εικόνα 5.1. Αποτέλεσμα εφαρμογής μετασχηματισμού καθρεπτισμό

5.2.2 Διαχωρισμός σε πλαίσια

Διαχωρισμός του κάτω τρίτου της εικόνας σε πλαίσια και ανακάτεμα των πλαισίων.

(a) Στιγμιότυπο της εικονικής κάμερας, χωρίς μετασχηματισμό.

(b) Στιγμιότυπο της εικονικής κάμερας, μετά τον μετασχηματισμό.

Εικόνα 5.2. Παράδειγμα εφαρμογής του μετασχηματισμού διαχωρισμού σε πλαίσια.

5.2.3 Μονόπλευρος Καθρεπτισμός

Αντανάκλαση μίας πλευράς του καρέ στην απέναντι. Το πρόγραμμα υποστηρίζει αντανάκλαση τοσο της δεξιάς όσο και της αριστερής πλευράς.

(a) Στιγμιότυπο της εικονικής κάμερας, μετά τον μετασχηματισμό εφαρμοσμένο στην αριστερή πλευρά του καρέ.

(b Στιγμιότυπο της εικονικής κάμερας, μετά τον μετασχηματισμό εφαρμοσμένο στη δεξιά πλευρά του καρέ.

Εικόνα 5.3. Αποτέλεσμα εφαρμογής του μετασχηματισμού μονόπλευρου καθρεπτισμού.

5.2.4 Χρωματική επικάλυψη

Επικάλυψη κάθε εικονοστοιχείου του καρέ με ένα συγκεκριμένο χρώμα.

Η επικάλυψη πραγματοποιείται σύμφωνα με τον τύπο:

$$r_{new} = min(r * (a/255) + r_{old} * (1 - a/255), 255) \quad (1)$$

$$g_{new} = min(g * (a/255) + g_{old} * (1 - a/255), 255) \quad (2)$$

$$b_{new} = min(b * (a/255) + b_{old} * (1 - a/255), 255) \quad (3)$$

Όπου η τετράδα (r, g, b, a) αντιπροσωπεύει το χρώμα επικάλυψης και την αδιαφάνειά του, η τριάδα $(r_{old}, g_{old}, b_{old})$ είναι το χρώμα του εικονοστοιχείου του δείγματος χωρίς επεξεργασία και η τριάδα $(r_{new}, g_{new}, b_{new})$ είναι το χρώμα του εικονοστοιχείου που παράγεται ως αποτέλεσμα.

Το πρόγραμμα υποστηρίζει την επιλογή του χρώματος, καθώς και της τιμής αδιαφάνειάς του, και το αποτέλεσμα είναι παρόμοιο με την προβολή του αρχικού καρέ μέσα από χρωματισμένο γυαλί.

Στιγμιότυπο της εικονικής κάμερας, μετά τη μίξη με χρώμα #23A9ED (●) αδιαφάνειας 125

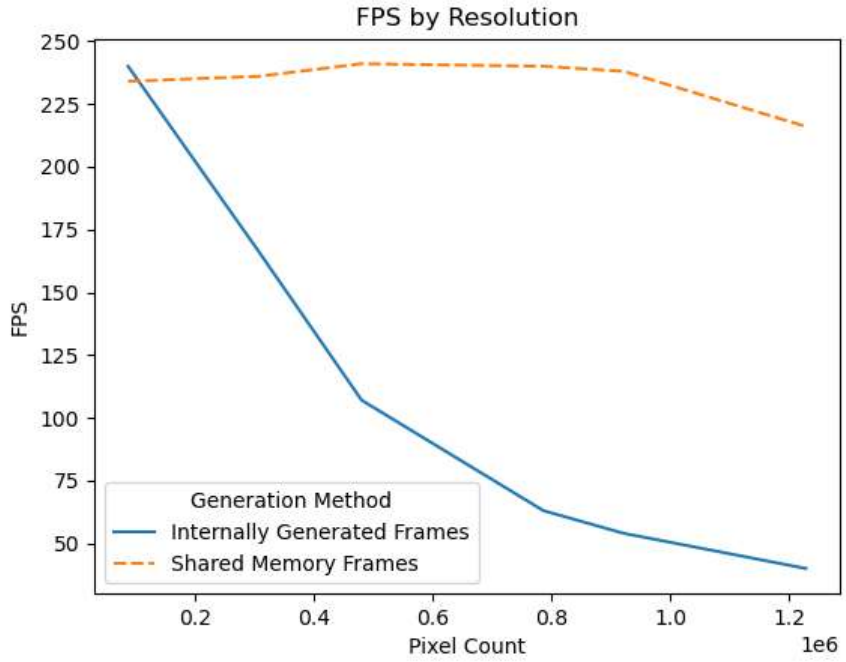
Στιγμιότυπο της εικονικής κάμερας, μετά τη μίξη με χρώμα #F2D487 (●) αδιαφάνειας 75.

Εικόνα 5.4. Αποτέλεσμα εφαρμογής του μετασχηματισμού χρώματος.

6 ΑΠΟΔΟΣΗ ΚΑΙ ΣΗΜΕΙΑ ΑΝΑΦΟΡΑΣ

6.1 Αξιολόγηση Απόδοσης

Η αξιολόγηση της εικονικής κάμερας χωρίστηκε σε δύο περιπτώσεις: στην περίπτωση που παράγονται από κάποιο εξωτερικό πρόγραμμα (πορτοκαλί γραμμή) και στην περίπτωση που τα καρέ παράγονται από την ίδια την εικονική κάμερα (μπλέ γραμμή). Κύριος άξονας σύγκρισης μεταξύ των δύο αυτών περιπτώσεων ήταν τα καρέ ανά δευτερόλεπτο (FPS), σε διάφορες αναλύσεις που προσφέρει η εικονική κάμερα.



Εικόνα 6.1. Γραφική απεικόνιση των FPS Εικονικής Κάμερας ανάλογα με τον αριθμό pixel στην ανάλυση της.

Παρατηρήθηκε ότι:

- Στην περίπτωση που τα καρέ παράγονται από την ίδια την κάμερα, η ανάλυσή της έχει κεντρικό ρόλο στις επιδόσεις της.
- Στην περίπτωση που τα καρέ παράγονται από εξωτερικό πρόγραμμα, οι επιδόσεις της κάμερας παραμένουν σχετικά σταθερές με τις μεταβολές της ανάλυσης.

Φυσικά τα παραπάνω αποτελέσματα οφείλονται στην μετακίνηση του φόρτου εργασίας από το πρόγραμμα της κάμερας σε κάποιο εξωτερικό, και τα καρέ που παράγονται από την κάμερα στην δεύτερη περίπτωση είναι συχνά ίδια καθώς το ίδιο καρέ παραμένει στην κοινή μνήμη μέχρι να αντικατασταθεί από το καινούριο. Ωστόσο, ο ρυθμός με τον οποίο η εικονική κάμερα διαβάζει και αποτυπώνει τα καρέ από την κοινή μνήμη δεν φαίνεται να επηρεάζεται σε σημαντικό βαθμό από την ανάλυσή της.

Επίσης ενδιαφέρον φαινόμενο αποτελεί ότι η απόδοση της εικονικής κάμερας μεγιστοποιείται περίπου στα 240 καρέ το δευτερόλεπτο και στις δύο

περιπτώσεις. Στην τιμή αυτή, ο παράγοντας της ανάλυσης φαίνεται να υποσκιάζεται από άλλες λειτουργίες που είναι απαραίτητες για την εικονική κάμερα.

6.2 Περαιτέρω Σημεία Ανάπτυξης

Μέσω της εργασίας, αναγνωρίστηκαν επίσης τα παρακάτω σημεία στα οποία η λειτουργικότητα της εικονικής κάμερας που αναπτύχθηκε μπορεί να επεκταθεί:

- Ορισμός δεδομένου ρυθμού καρέ ανά δευτερόλεπτο, ενσωματωμένος στο πρόγραμμα της εικονικής κάμερας.
- Απεικόνιση ροής βίντεο σε πολλαπλούς τύπους δεδομένων, όχι μόνο RGB.
- Απόλυτα δυναμική αλλαγή ανάλυσης, όσο δηλαδή η εικονική κάμερα προβάλλει δεδομένα.

Στην τρέχουσα υλοποίηση της κάμερας, τα πρώτα δύο σημεία θεωρούνται αρμοδιότητες των προγραμμάτων καταγραφής καρέ και όχι της ίδιας της εικονικής κάμερας.

7 ΜΕΛΕΤΗ ΕΙΚΟΝΙΚΩΝ ΜΙΚΡΟΦΩΝΩΝ

Στο τελευταίο μέρος της εργασίας εξετάστηκε η ανάπτυξη εικονικών μικροφώνων. Ενδεικτικά, αναλύθηκε ο πηγαίος κώδικας του εργαλείου Virtual Audio Driver και μελετήθηκαν οι λειτουργίες του.

Το σύστημα DirectShow δεν προσφέρει την ίδια δυνατότητα για ανάπτυξη εικονικών συσκευών ήχου και δεν υπάρχουν παρόμοια εργαλεία ενσωματωμένα στο λειτουργικό σύστημα Windows για την διευκόλυνση της παραγωγής τους. Για αυτόν τον λόγο, η ανάπτυξη εικονικών συσκευών ήχου βασίζεται συχνότερα σε προγράμματα οδήγησης πυρήνα, γεγονός που αυξάνει την πολυπλοκότητα αλλά προσφέρει μεγαλύτερη ενσωμάτωση με το σύστημα. Μία αρχιτεκτονική που προσφέρεται για τα προγράμματα αυτά είναι η Miniport Driver, η οποία επιτρέπει στον προγραμματιστή να ασχοληθεί κυρίως με την υλοποίηση

θεμάτων συγκεκριμένων για τη συσκευή του και να βασίζεται σε ήδη υλοποιημένα γενικά προγράμματα οδήγησης για άλλες λειτουργίες [5].

Με την υλοποίηση διεπαφών που προσφέρει το λειτουργικό σύστημα, το πρόγραμμα παρέχει στον υπολογιστή τις απαραίτητες πληροφορίες για την αναγνώριση της εικονικής συσκευής και επιτρέπει στο σύστημα να της στείλει αιτήματα. Εσωτερικά, τα αιτήματα διεκπεραιώνονται προσομοιώνοντας τις λειτουργίες μίας πραγματικής συσκευής ήχου. Η παραπάνω διαδικασία περιλαμβάνει τον ορισμό εικονικών κυκλωμάτων για τα εικονικά ηχεία και το εικονικό μικρόφωνο, των χειριστηρίων της έντασης και του μίκτη, καθώς και υλοποιήσεις αλγορίθμων με σκοπό την παραγωγή ήχου, μεταξύ άλλων.

8 ΣΥΜΠΕΡΑΣΜΑΤΑ

Η εργασία απέδειξε ότι είναι εφικτή η ανάπτυξη μίας λειτουργικής εικονικής κάμερας σε Windows χωρίς άμεση παρέμβαση σε επίπεδο πυρήνα, αξιοποιώντας τη βιβλιοθήκη DirectShow.

Η χρήση κοινής μνήμης ως δίαυλος καρέ, μέσα στον οποίο μπορούν εξωτερικά προγράμματα να καταγράψουν δεδομένα, επέτρεψε τον διαχωρισμό αρμοδιοτήτων και τη δημιουργία επεκτάσιμης αρχιτεκτονικής. Η απόδοση κρίθηκε ικανοποιητική για εφαρμογές πραγματικού χρόνου.

Η μελέτη των εικονικών μικροφώνων ανέδειξε τη σημασία και τη λειτουργικότητα των kernel-mode drivers σε πιο σύνθετες περιπτώσεις. Καταλήξαμε όμως στο συμπέρασμα ότι στη γενική περίπτωση η ανάπτυξη εικονικών συσκευών διεκπεραιώνεται με μεγαλύτερη ευκολία όταν χρησιμοποιούνται τεχνολογίες ανάπτυξης εφαρμογών σε τομέα σχετικό με το είδος της εικονικής συσκευής, εφόσον αυτές οι τεχνολογίες αποτελούν υπαρκτές και έμπιστες επιλογές.

9 ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Πιθανές κατευθύνσεις μελλοντικής έρευνας πάνω σε εικονικές συσκευές περιλαμβάνουν:

- Ανάπτυξη εικονικών συσκευών εισόδου με σκοπό τον καθορισμό και τη διευκόλυνση εναλλακτικών χρήσεων του υπολογιστή.
- Αναγνώριση και μεταβολή χαρακτηριστικών προσώπου για υλοποίηση πιο σύνθετων μετασχηματισμών.
- Σύγκριση απόδοσης προγραμμάτων εφαρμογής μετασχηματισμών σε βίντεο ζωντανής ροής, ιδιαίτερα σε διαφορετικά πλαίσια υλοποίησης (διαφορετικά εργαλεία ή και γλώσσες προγραμματισμού).
- Ανάπτυξη προγραμμάτων οδήγησης συσκευής για διαφορετικά λειτουργικά συστήματα ή και ανεξάρτητα προγράμματα οδήγησης που λειτουργούν σε κάθε βασικό λειτουργικό σύστημα.

ΑΝΑΦΟΡΕΣ

[1] Andrew Kent Warfield. Virtual Devices for Virtual Machines. PhD thesis, University of Cambridge, 2006. URL: https://docslib.org/doc/5563323/virtual-devices-for-virtual-machines-andrew-kent-warfield.

[2] T. He and A.E. Kaufman. Virtual input devices for 3d systems. In Proceedings Visualization ’93, 1993. doi:10.1109/VISUAL.1993.398862.

[3] Microsoft. About Direct Show Filters, 2023. URL: https://learn.microsoft.com/en-us/windows/win32/directshow/about-directshow-filters.

[4] Microsoft. Pin Connections, 2023. URL: https://learn.microsoft.com/en-us/windows/win32/directshow/pin-connections.

[5] Microsoft. Minidrivers, Miniport drivers, and driver pairs, 2024. URL: https://learn.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/minidrivers-and-driver-pairs

Colored Huge Pages: A Hardware-Software Approach for Enhanced Isolation and Performance

Georgios - Alexandros Kostas

ABSTRACT

Multicore CPUs typically share the Last-Level Cache (LLC) across cores, leading to interference between co-executing workloads with significant performance and security implications. Page coloring has emerged as an effective software mechanism for LLC partitioning. Simultaneously, virtual memory enables fundamental abstractions, but incurs increasing performance overheads due to address translation. Huge pages alleviate this issue by expanding TLB reach, thereby reducing TLB misses and the associated costly page table walks. However, these two techniques are considered mutually exclusive, since huge pages span all LLC sets, precluding coloring.

This thesis introduces Colored Huge Pages (CHP), a hardware-software co-design that enables the simultaneous use of page coloring and huge pages. By distributing the physical frames of a virtually contiguous huge page across physical memory in a predictable strided pattern, our design allows coloring of the individual pages while preserving the TLB reach and translation efficiency of conventional huge pages. On the software side, we modify the OS allocator to construct colored huge pages by extracting appropriately colored pages from larger physical blocks and caching leftover mappings for future use. On the hardware side, we extend the L2 TLB to efficiently translate these mappings by leveraging their regular structure. We implement our approach in a recent Linux kernel and evaluate it using memory intensive workloads. CHP mitigates LLC contention and address translation overheads, improving performance by 33.7% compared to using 4 KB pages without LLC partitioning, requiring only minimal OS and architectural modifications. Contrary to prior approaches, our proposal

maintains comparable effectiveness under fragmentation, avoids inducing additional cache misses, and incurs only negligible page fault overhead relative to Transparent Huge Pages (THP).

Keywords: Page Coloring, Huge Pages, Cache Contention, Last-Level Cache, Virtual Memory

ADVISOR

Vasileios Karakostas, Assistant Professor, NKUA

1 INTRODUCTION

Modern multiprocessing systems share several microarchitectural components, such as the Last Level Cache (LLC), across multiple cores, making them vulnerable to contention and interference. This can lead to significant performance degradation, reduced predictability, and increased security risks. *Page coloring* [15, 26] is a memory management technique that allows the operating system (OS) to control which portions of shared hardware resources a process can access by carefully managing the allocation of its physical pages, in order to improve performance isolation and enhance system security.

Simultaneously, virtual memory has become a fundamental abstraction in modern computing systems, enabling process isolation and memory protection by decoupling the virtual address space of different programs from the underlying physical memory layout. To realize this abstraction, virtual addresses are mapped to physical addresses via multi-level page tables managed by the operating system, while hardware support transparently walks the page tables and caches mappings in the Translation Lookaside Buffer (TLB). However, these address translation mechanisms incur significant performance overheads, requiring costly page table walks on every TLB miss, especially in modern memory-intensive workloads. *Huge pages* [10, 18] are often employed to reduce the frequency of TLB

misses by expanding TLB reach, thereby effectively reducing address translation costs.

Page coloring and huge pages are traditionally considered incompatible, because huge pages occupy all available colors, leaving no address bits that influence cache indexing under OS control [7, 8]. Prior work aiming to combine the two techniques suffers either from significant contiguity constraints on the OS, limiting address translation effectiveness under memory fragmentation [7], or restricts every huge page to a specific color, unnecessarily increasing conflict cache misses and harming performance [8]. In addition, those prior studies did not present in detail, nor evaluate, the necessary OS support for combining page coloring with huge pages. Our goal in this thesis is to enable the simultaneous use of huge pages and page coloring for LLC partitioning, while retaining the performance advantages and surpassing the limitations of prior approaches through a practical design.

We propose *Colored Huge Pages* (CHP): a hardware-software co-design that enables the simultaneous use of page coloring and huge pages by breaking virtually contiguous huge pages into multiple colored physically non-contiguous base pages arranged in a predictable, strided pattern. Through lightweight microarchitectural enhancements and OS support, the system enables cache partitioning, while retaining the TLB reach benefits of traditional huge pages by translating colored huge pages efficiently, as if they were physically contiguous. Our design includes a custom allocator, integrated within the Linux kernel, that constructs these strided memory mappings, along with lightweight modifications in the L2 TLB, Page Table, and Page Table Walker that exploit the regular mapping structure to accelerate address translation.

To the best of our knowledge, this is the first work to propose novel hardware and system-level extensions to significantly relax the strict memory contiguity requirements imposed on the OS allocator for the construction of colored huge pages, which become impractical in the presence of memory fragmentation under realistic system conditions. Furthermore, this is the first work to design, implement, and evaluate the operating system support required to integrate page coloring with huge pages.

We prototype CHP in Linux kernel v6.6 and evaluate our solution using a set of memory-intensive workloads and an analytical performance model. CHP enables the simultaneous use of page coloring and huge pages, improving performance by 33.7% - compared to 17.4% with 4 KB coloring and 21% with Transparent Huge Pages (THP). CHP maintains robustness comparable to THP under memory fragmentation and delivers lower cumulative page fault overhead than both default paging and regular 4 KB coloring. Thanks to its flexibility, our design achieves higher effectiveness under memory fragmentation compared to [7] and outperforms [8] by up to 13%.

In summary, the key contributions of this thesis are:

- We quantify the impact of cache interference and address translation and show that combining page coloring with huge pages can simultaneously reduce LLC interference and mitigate TLB pressure.
- We propose Colored Huge Pages (CHP): a complete hardware-software co-design that enables a huge page to be mapped to individually non-contiguous colored base pages while preserving translation efficiency.
- We comprehensively evaluate the combined benefits of CHP under a variety of system conditions, including memory pressure and fragmentation scenarios.

2 MOTIVATION

2.1 Page Coloring & Huge Pages

Page coloring has emerged as a practical technique for software-based partitioning of shared hardware resources [15, 19, 20, 22, 6, 33, 29, 31]. In this work, we focus on page coloring for LLC partitioning. Page coloring is orthogonal to hardware-based way cache partitioning techniques [2, 12, 3, 13]. Those approaches only enable coarser-grained partitions, since the number of ways in modern caches is severely restricted by latency, energy consumption, and die area tradeoffs. As a result, the number of partitions does not scale well with the

increasing core counts of contemporary systems. Unlike way-based partitioning, page coloring applies set partitioning, enabling finer control than is otherwise achievable through either technique in isolation [28]. Moreover, it remains the only viable mechanism for controlling the allocation of resources that lack dedicated hardware partitioning support.

Independently, virtual memory and address translation through the TLB have become a major performance bottleneck in modern memory-intensive workloads. Huge pages [10, 18] can help mitigate this issue by mapping a significantly larger portion of a process's address space with a single translation entry, i.e., 2 MB vs 4 KB on x86 systems. This increases TLB reach and reduces the frequency of TLB misses, which would otherwise trigger expensive page table walks. Furthermore, huge pages shorten the walk by skipping the last page table level. Overall, huge pages significantly improve application performance.

2.2 Potential Performance Improvement

Page coloring and huge pages are incompatible, because the large page offset in a huge page usually consumes all the address bits, leaving none of the cache-indexing bits under OS control [7, 8]. However, combining the two techniques would achieve significant performance benefits by reducing cache contention and the number of page table walks.

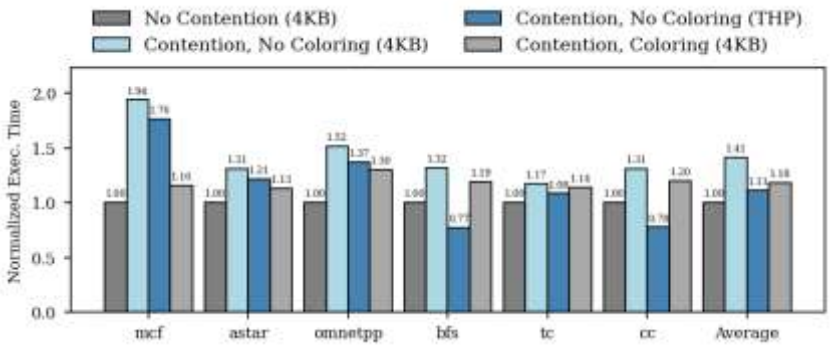


Fig. 1: Normalized execution time under contention with 4 KB pages, THP, and page coloring, compared to no contention.

To motivate our approach, we study the performance impact of cache contention, huge pages, and page coloring. Section 4 describes the evaluation methodology.

Fig. 1 shows the normalized execution time relative to the baseline (no contention, 4 KB pages). Cache contention degrades performance by up to 1.94× (mcf) and 1.41× on average. Page coloring [32] alleviates this interference, reducing the average overhead to 1.18×. THP [10] provides additional performance gains, particularly in bfs and cc, where the execution time under contention improves to 0.77× and 0.78× of the baseline. These results indicate that combining huge pages with cache isolation via page coloring can deliver substantially greater performance improvements than either technique in isolation.

2.3 Prior Approaches

Only two prior studies have investigated the integration of page coloring with huge pages. Table 1 summarizes how CHP compares to the previous approaches in several key aspects.

Table 1: Comparison of huge page coloring approaches.

Approach	Fragmentation Robustness	No Extra Cache Conflicts	HW/SW Compatibility	OS Design & Evaluation
Scattered [7]	✗	✓	✓	✗
SWAP [8]	✓	✗	✗	✗
CHP	✓	✓	✓	✓

Scattered Superpages [7] map a virtual huge page across multiple physical huge pages. Similar to CHP, that approach requires expanding the TLB to store additional color information per huge page. However, Scattered Superpages are constructed using a contiguous set of *multiple physical huge pages*. That memory management approach limits the resilience and effectiveness of Scattered Superpages under realistic fragmentation conditions (Section 5.3). Motivated by this limitation, we introduce novel hardware and OS mechanisms that allow each colored huge page to consist of multiple independent sub-mappings, thereby relaxing contiguity requirements.

Another prior work [8] integrates huge pages with page coloring by swapping bits from the huge page number with LLC index bits prior to cache access,

creating a *pseudo-physical address space*. However, that design restricts each huge page to a single color, increasing conflict misses and degrading performance in some benchmarks (Section 5.4). Moreover, this hardware indirection hides real physical memory from the OS, which impedes compatibility with existing software and hardware components (e.g., drivers and DMA engines) that require real physically contiguous memory. Supporting both real and pseudo-physical spaces significantly complicates OS management. In contrast, CHP avoids inducing additional conflict misses by distributing portions of huge pages to multiple colors and allows the OS to keep managing the real physical address space, providing full compatibility with existing mechanisms.

Finally, neither of the two prior approaches design, implement, nor evaluate the operating system support required to integrate page coloring with huge pages.

3 COLORED HUGE PAGES

The *key idea* of our approach is that a huge page can remain contiguous in virtual memory, but get broken down into multiple individual non-contiguous physical frames, allocated with a regular pattern. With lightweight microarchitectural modifications, the hardware exploits this allocation pattern to perform address translation efficiently, just as if the pages were physically contiguous, preserving the TLB reach and performance benefits of huge pages.

We propose *Colored Huge Pages* (CHP): a complete system design that spans the hardware-software boundary, combining the necessary microarchitectural modifications with operating system support. We assume and maintain compatibility with x86 translation mechanisms in this text, but the principles can be trivially extended to other ISAs.

3.1 Overview

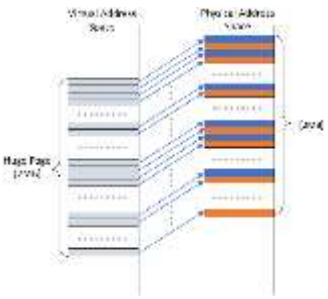


Fig. 2: Regular Huge Page (no coloring)

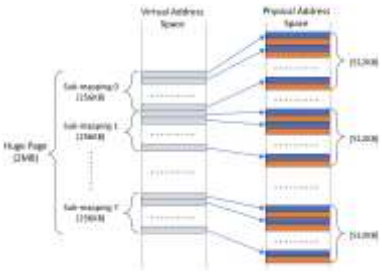


Fig. 3: Colored Huge Page with 1 out of 2 colors

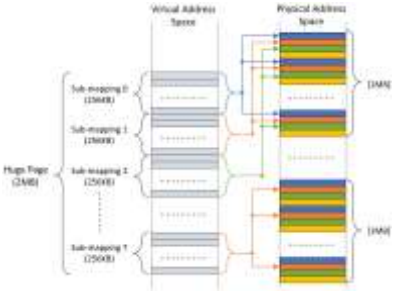


Fig. 4: Colored Huge Page with 3 out of 4 colors

CHP targets large anonymous Virtual Memory Areas (VMAs) of memory-intensive processes. Page faults within these VMAs are intercepted and handled by a custom page allocation system, which backs each faulting 2 MB region with carefully constructed mappings. These mappings consist of base pages allocated at a constant stride so that they share the same color(s). Pages with colors differing from the target one are effectively skipped, resulting in mappings composed exclusively of pages with the desired color(s), as illustrated in Fig. 3. The regularity of these mappings allows the offset of the i-th page from the start of the mapping to be computed with inexpensive bitwise operations, enabling accelerated address translation with minimal microarchitectural modifications, even though the individual pages are not physically contiguous.

To construct a colored huge page, our custom allocator first identifies a sufficiently large contiguous memory region, from which base pages corresponding to the desired colors are selectively extracted. On its own, such an approach is impractical due to the memory fragmentation exhibited in long-running systems [30]. To alleviate these strict contiguity requirements, we propose a novel approach that, with minimal additional hardware support and negligible overhead, allows each 2 MB strided mapping to be divided into 8 independent sub-mappings. Each of these sub-mappings maintains the previously described strided structure, but can begin independently of the others, significantly reducing the demand for physically contiguous memory regions, while also improving color allocation flexibility when multiple colors are

permitted for a process. An example of this subdivision is illustrated in Fig. 4. CHP falls back to existing mechanisms if a strictly compliant region cannot be allocated and operates completely transparently to the applications in a best-effort manner.

3.2 Hardware Modifications

3.2.1 Modified L2 TLB and Address Translation

Our design leaves the L1 TLB path intact and only extends the L2 TLB. By integrating CHP into the L2 TLB, we preserve the reduced page table walk frequency of huge pages while avoiding added latency in the L1 TLB's critical path. CHP augments each L2 TLB entry with the starting Physical Frame Numbers (PFNs) of all eight sub-mappings, widening it to include seven additional PFNs. Tag, permission and access bits are shared across all sub-regions. A dedicated bit per L2 TLB entry indicates whether it refers to a conventional (huge) page translation or a colored huge page. The extended L2 TLB is queried after an L1 TLB miss. Upon an L2 TLB hit for a colored huge page, the corresponding 4 KB entry is constructed and cached into the L1 TLB to accelerate subsequent accesses. Importantly, L2 TLB indexing remains unaffected, as the number of entries remains unchanged.

Fig. 5 illustrates how the proposed translation mechanism for the augmented L2 TLB is realized in hardware to achieve fast address translation for colored huge pages, assuming 4 KB and 2 MB base and huge page sizes:

1) **VPN Extraction**: the lower 21 offset bits of the incoming virtual address are discarded to extract the Virtual (Huge) Page Number (VPN).
2) **Parallel TLB Lookup:** the VPN is looked up in the enhanced L2 TLB as normal. A dedicated per-entry *CHP bit* identifies TLB hits corresponding to colored huge pages.
3) **Sub-Entry Index Calculation**: upon a colored huge page hit, the three most significant bits of the 2 MB page offset (bits 18–20) are isolated to determine which sub-mapping corresponds to the current address.
4) **Sub-Entry Extraction**: the selected sub-entry is extracted based on the computed index. Page access permissions are validated at this stage and the base PFN of the relevant sub-mapping is obtained.

5) **Page Offset Computation**: the offset of the target page compared to the start of the corresponding sub-mapping is calculated by multiplying its page index with the constant number of available colors. This operation can be performed using a bitwise shift.
6) **Final Target PFN Calculation**: the final PFN of the target page is derived by bitwise *OR*-ing the offset from Step 5 to the base PFN from Step 4.

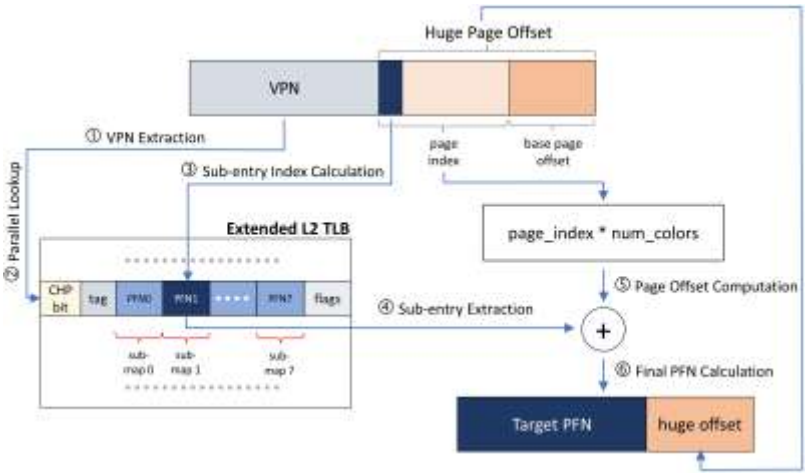


Fig. 5: Hardware-assisted address translation for colored huge pages. *Page index* refers to the offset of a 4 KB base page within its surrounding 2 MB huge page.

3.2.2 Modified Page Table & Page Table Walk

Upon an L2 TLB miss within a colored huge page, 8 PFNs - corresponding to the 8 individual PTEs of the sub-mappings - must be fetched into the augmented L2 TLB. Fortunately, this incurs no additional cost in the page table walk path, as all 8 PTEs lie within a single cache line and can be retrieved with a single memory reference [27].

We modify the Page Table Walker (PTW) so that upon reaching a PMD entry with the *Colored-Huge-Page Bit* set (managed by the OS), it fetches 8 PTEs from the address indicated by the PMD into the L2 TLB. The PTW identifies the sub-entry for the faulting address, fetching and using it immediately while filling the rest in the background.

3.3 Operating System Support

In CHP, the OS must allocate, manage, and install mappings in the proposed strided format. We prototype our design on Linux Kernel v6.6, introducing a dedicated colored huge page allocator built on top of Linux's buddy allocator [9]. This approach preserves compatibility with existing memory management while enabling color-aware huge page allocation.

CHP operates transparently to applications, intercepting anonymous page faults within large VMAs and attempting to populate the faulting 2 MB region with a colored huge page. Constructing such a page requires locating eight independent 256 KB strided sub-mappings, with all pages within each sharing the same color. To allocate each sub-mapping, CHP requests from the buddy allocator a sufficiently large (256 KB × number of system colors) contiguous block from which it extracts the desired strided mappings.

If successful, the required individual pages are carved out from the contiguous block and installed into the process's address space by crafting the appropriate Page Table Entries (PTEs). To enable prototyping on existing hardware, we currently register and install each 4 KB page individually. An actual implementation would install only one PTE per sub-mapping (8 total). The OS also sets the special *Colored-Huge-Page Bit* at the PMD level to signal to the PTW that this region is backed by a colored huge page.

CHP additionally introduces an *Allocator Cache* for caching the left-over physical pages from the allocated block for future use. Each sub-mapping request first attempts to get serviced by the Allocator Cache, before allocating pages from a new contiguous physical block obtained through the buddy allocator. Additionally, strided mappings can be reclaimed from the process's memory and returned to the Allocator Cache when a VMA is unmapped. The CHP allocator operates in a best-effort manner, falling back to 4 KB page coloring if the desired strided mapping cannot be constructed. Contrary to SWAP [8], CHP ensures fair color distribution by assigning each sub-mapping a different color from the process's allowed set in a round-robin manner.

4 METHODOLOGY

We evaluate CHP on an 8-core Intel i7-10700 machine with 16 GB RAM, and prototype our implementation in Linux Kernel v6.6. For the evaluation we use benchmarks from GAPBS [5] and SPEC 2006 [24]. Cache contention is introduced via *stress-ng* [25]. Cache isolation for the stressor is enforced using regular page coloring [32]. Each benchmark is assigned the minimum number of colors that keeps its runtime within 4% with respect to using the entire LLC when running alone. The remaining colors are assigned to the stressor. We pin applications to separate cores and present average results from 3 runs.

Our evaluation focuses on performance and relies on an analytical model driven by hardware performance counters to estimate the benefits of CHP. While simulation could model the proposed translation hardware support, it would be prohibitively slow for practical evaluation [4, 14, 1]. We report runtimes under four scenarios: (i) contention with 4 KB pages (baseline), (ii) THP without coloring, (iii) 4 KB page coloring and (iv) projected runtime with CHP (huge pages + coloring).

Using Linux *perf*, we collect total execution cycles (T) and cycles spent in page table walks (C). We decompose all reported runtimes (R) into translation ($\mathrm{O}_{\mathrm{trans}}$) and cache ($O_{cache}$) components, such that the total runtime is given by: $R = O_{trans} + O_{cache}$. All runtimes are normalized to the 4 KB contention baseline. For each execution scenario S, we compute the normalized translation component as $O^{S}_{trans} = \frac{C_S}{T_{contention-4K}}$ and the cache component as $O^{S}_{cache} = \frac{T_s - C_S}{T_{contention-4K}}$.

To estimate CHP performance, we assume its translation cost equals that of contention-THP. We model the effect of coloring-induced isolation in the execution time by scaling the contention-THP cache component by the cache speedup of 4 KB coloring. This pessimistic model likely underestimates CHP's benefits, as it ignores reductions in cache pressure from fewer TLB misses. The overall CHP runtime estimate (R_{CHP}) is given by (1).

$$R_{CHP} = \underbrace{O^{cont\text{-}THP}_{trans}}_{O^{CHP}_{trans}} + \underbrace{O^{cont\text{-}THP}_{cache} \cdot \frac{O^{color\text{-}4K}_{cache}}{O^{cont\text{-}4K}_{cache}}}_{O^{CHP}_{cache}} \quad (1)$$

5 EVALUATION

5.1 Performance

Fig. 6 presents the runtime decomposition for all benchmarks under the four scenarios mentioned in Section 4, with all runtimes normalized to the 4 KB contention baseline. Each bar decomposes total runtime into address translation and cache components.

When used individually, THP and page coloring reduce runtime to 79% (as low as 57.6%) and 82.6% (as low as 63%) of the baseline, respectively. Even assuming no additional cache benefits from reduced TLB misses and page table walks beyond what coloring with 4 KB pages provides, our approach consistently outperforms both THP and 4 KB coloring, reducing average runtime to 66.3% of the baseline and reaching as low as 54.4% for cc.

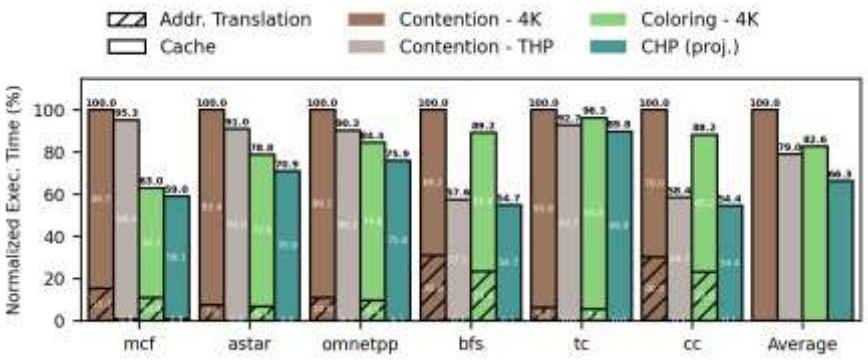


Fig. 6: Normalized runtime breakdown showing projected performance gains from CHP, compared to 4 KB pages, THP and 4 KB page coloring, all under contention.

5.2 Fragmentation Sensitivity

We evaluate the success ratio of CHP allocations under varying memory fragmentation levels. Fragmentation is quantified by the *fragmentation index* (F_i), adapted from [11], which denotes the fraction of physical memory that cannot be backed by the currently available huge pages. We focus on mcf and bfs, whose memory footprints correspond to roughly 11% and 29% of the system's physical memory. Therefore, assuming regular huge pages (e.g., THP), the critical fragmentation threshold, beyond which allocation success begins to degrade, is expected at $\mathrm{F_i} \approx 0.89$ for mcf and $\mathrm{F_i} \approx 0.71$ for bfs.

As shown in Fig. 7, our solution is resilient under low to moderate fragmentation. For both benchmarks, success ratios degrade before the critical fragmentation point, as much as 10% earlier for bfs, indicating CHP's slightly higher fragmentation sensitivity relative to THP. Nonetheless, allocations remain near-perfect up to moderately high fragmentation ($F_i = 0.6$ even for the larger workload), demonstrating reasonable robustness relative to THP under realistic fragmentation conditions.

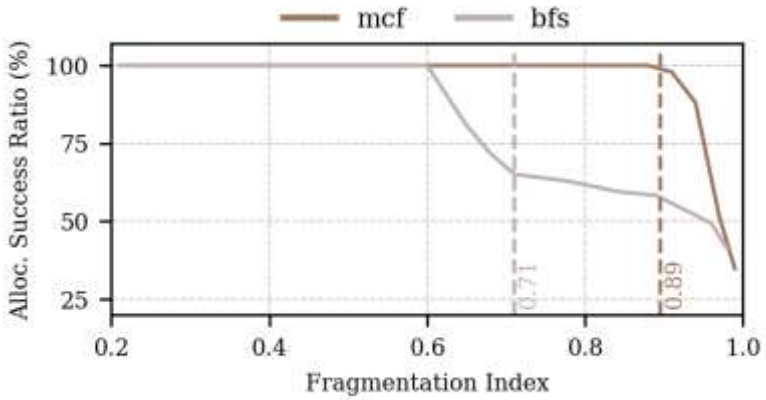


Fig. 7: CHP allocation success ratio vs. fragmentation index.

5.3 Number of Sub-mappings

Fig. 8 shows allocation success ratios for 1, 2, 4, and 8 sub-mappings across different fragmentation levels. Using only a single sub-mapping corresponds to Scattered Superpages [7], where the strided mapping spans *strictly contiguous* physical huge pages. We use bfs for this analysis due to its large 4.5 GB footprint.

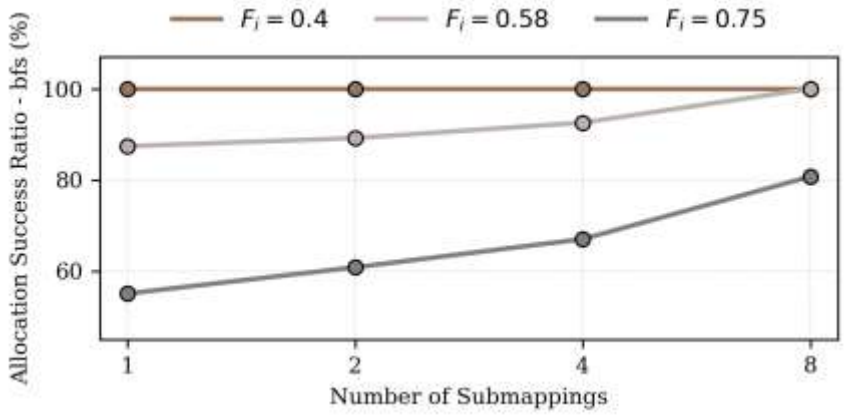


Fig. 8: Impact of fragmentation and sub-mapping counts on bfs allocation success ratio.

As fragmentation approaches the critical threshold ($F_i = 0.58$), using 8 sub-mappings preserves a 100% allocation success ratio, whereas fewer sub-mappings degrade allocation effectiveness. This effect becomes more pronounced at higher fragmentation levels ($F_i = 0.75$), where reducing the number of sub-mappings leads to even more substantial drops in successful allocations. These results demonstrate that the finer sub-mapping granularity enabled by our design significantly improves resilience to fragmentation compared to Scattered Superpages under realistic memory conditions.

5.4 Comparison to SWAP

We modify our implementation to emulate the mappings created by SWAP, which contrary to CHP restricts an entire huge page to a single cache color, and measure the performance difference compared to CHP. For the *GAPBS* workloads, we notice only negligible differences between the two approaches. However, SWAP increases LLC misses for mcf by 9.4% and 12% when using 5 and all 8 colors, leading to performance degradations of 11.8% and 13.1% respectively. This analysis demonstrates that the flexibility provided by CHP can significantly outperform SWAP in some cases, effectively offsetting its modest additional overheads.

5.5 Overhead Analysis

We measure average and tail (99th percentile) page fault latency, fault counts, and cumulative fault time for bfs. CHP introduces overhead to the fault path compared to 4 KB pages, 4 KB coloring using PALLOC [32], and THP. Unlike THP, which installs a huge page via a single PMD update, our implementation must register and install each 4 KB page individually to allow prototyping on existing hardware. Nonetheless, CHP reduces fault frequency to THP levels, lowering cumulative fault time well below both 4 KB paging and PALLOC. Ultimately, the few hundred milliseconds of page fault overhead (< 0.2% of bfs's runtime) are negligible relative to workload duration (hundreds of seconds) and outweighed by the benefits of reduced cache contention.

In terms of area, CHP expands the data part of each L2 TLB entry by ~8x to store the additional PFNs. However, the tag and permission bits, along with the number of entries remain unchanged, so no extra hardware is required for indexing. This minimal hardware overhead is outweighed by the increased TLB reach (by 512×) enabled by each augmented entry.

6 RELATED WORK

Sections 2 and 5 compared CHP qualitatively and quantitatively with the most closely related work [7, 8]. Here we discuss other prior page coloring work for cache partitioning. Page coloring was first proposed by Taylor et al. [26] to improve TLB hit rate, while Kessler et al. [15] later proposed page placement algorithms for cache partitioning. Since then, two main software approaches have emerged. In static page coloring [15, 19, 20, 22], a process is allocated a portion of the cache statically during its execution. In contrast, dynamic coloring [19, 6, 33, 29, 31] adapts to time-varying cache access patterns and allocation needs by relying on page recoloring support. In addition, previous efforts [22, 29, 23, 21, 17] have also applied page coloring in cloud environments to improve inter-VM isolation. Finally, Wang et al. [28] combined page coloring with hardware way partitioning to support fine-grained cache partitions.

7 CONCLUSIONS

Modern systems can benefit significantly from the simultaneous use of page coloring and huge pages to improve both cache isolation and translation efficiency. In this work we proposed CHP, a custom hardware-software co-design that enables virtually contiguous huge pages to be distributed across individually colored physical frames without compromising address translation performance. We designed and implemented operating system support in the Linux kernel and proposed lightweight microarchitectural enhancements to support fast translation of these mappings, while simultaneously relaxing the contiguity constraints on the OS memory allocator. Our evaluation results showed that CHP successfully manages to combine the benefits of both page coloring and huge pages, providing improved performance with respect to prior work.

This BSc thesis has resulted in a publication at the 2026 Design, Automation, and Test in Europe Conference (DATE 2026).
Full reference: Georgios-Alexandros Kostas, Dimitris Gizopoulos, Vasileios Karakostas, “Colored Huge Pages: A Hardware-Software Approach for Enhanced Isolation and Performance”, in Proceedings of the 2026 Design, Automation, and Test in Europe Conference (DATE), Verona, Italy, April 2026 [16].

REFERENCES

[1] Chloe Alverti, Stratos Psomadakis, Vasileios Karakostas, Jayneel Gandhi, Konstantinos Nikas, Georgios Goumas, and Nectarios Koziris. Enhancing and Exploiting Contiguity for Fast Memory Virtualization. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

[2] AMD64 Technology Platform Quality of Service Extensions. https://www.amd.com/-content/dam/amd/en/documents/processor-tech-docs/other/56375_1_03_PUB.pdf.

[3] Arm DynamIQ Shared Unit Technical Reference Manual. https://-developer.arm.com/documentation/100453/0401/L3-cache/L3-cache-partitioning.

[4] Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, and Michael M. Swift. Efficient virtual memory for big memory servers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, page 237-248, 2013.

[5] Scott Beamer, Krste Asanovic, and David Patterson. The GAP Benchmark Suite. a*rXiv preprint arXiv:1508.03619*, 2015.

[6] Brian N Bershad, Dennis Lee, Theodore H Romer, and J Bradley Chen. Avoiding Conflict Misses Dynamically in Large Direct-Mapped Caches. In *Proceedings of the sixth international conference on Architectural support for programming languages and operating systems*, pages 158–170, 1994.

[7] Licheng Chen, Yanan Wang, Zehan Cui, Yongbing Huang, Yungang Bao, and Mingyu Chen. Scattered Superpage: A Case for Bridging the Gap between Superpage and Page Coloring. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 177–184, 2013.

[8] Zehan Cui, Licheng Chen, Yungang Bao, and Mingyu Chen. A Swap-based Cache Set Index Scheme to Leverage both Superpage and Page Coloring Optimizations. In *51st Annual Design Automation Conference (DAC)*, page 1-6, 2014.

[9] M Gorman. Understanding The Linux Virtual Memory Manager. https://-www.kernel.org/doc/gorman/html/understand/.

[10] Mel Gorman and Andrea Arcangeli. Transparent Hugepages in the Linux Kernel. https://www.kernel.org/doc/Documentation/vm/transhuge.txt.

[11] Mel Gorman and Andy Whitcroft. The What, The Why and the Where To of Anti-Fragmentation. In *Ottawa Linux Symposium*, volume 1, pages 369–384. Citeseer, 2006.

[12] Andrew Herdrich, Edwin Verplanke, Priya Autee, Ramesh Illikkal, Chris Gianos, Ronak Singhal, and Ravi Iyer. Cache QoS: From concept to reality in the Intel Xeon processor E5-2600 v3 product family. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 657–668. IEEE, 2016.

[13] Introduction to Cache Allocation Technology in the Intel® Xeon® Processor E5 v4 Family. https://www.intel.com/content/www/us/en/developer/articles/technical/-introduction-to-cache-allocation-technology.html.

[14] Vasileios Karakostas, Jayneel Gandhi, Furkan Ayar, Adrián Cristal, Mark D. Hill, Kathryn S. McKinley, Mario Nemirovsky, Michael M. Swift, and Osman Ünsal. Redundant Memory Mappings for Fast Access to Large Memories. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, ISCA '15, page 66-78, 2015.

[15] Richard E Kessler and Mark D Hill. Page Placement Algorithms for Large Real-Indexed Caches. *ACM Transactions on Computer Systems (TOCS)*, 10(4):338–359, 1992.

[16] Georgios-Alexandros Kostas, Dimitris Gizopoulos, and Vasileios Karakostas. Colored Huge Pages: A Hardware-Software Approach for Enhanced Isolation and Performance. In *Proceedings of the 2026 Design, Automation, and Test in Europe Conference (DATE)*, Verona, Italy, April 2026.

[17] Haifeng Li, Tianyue Lu, Yuhang Liu, and Mingyu Chen. Make Page Coloring More Efficient on Slice-Based Three-Level Cache. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 310–317. IEEE, 2019.

[18] libhugetlbfs(7) - Linux man page. https://linux.die.net/man/7/libhugetlbfs.

[19] William L Lynch, Brian K Bray, and Michael J Flynn. The Effect of Page Allocation on Caches. *ACM SIGMICRO Newsletter*, pages 222–225, 1992.

[20] Swann Perarnau, Marc Tchiboukdjian, and Guillaume Huard. Controlling cache utilization of hpc applications. In *Proceedings of the International Conference on Supercomputing*, page 295-304, 2011.

[21] Alberto Scolari, Davide Basilio Bartolini, and Marco Domenico Santambrogio. A Software Cache Partitioning System for Hash-Based Caches. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2016.

[22] Alberto Scolari, Filippo Sironi, Davide B. Bartolini, Donatella Sciuto, and Marco D. Santambrogio. Coloring the Cloud for Predictable Performance. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, 2013.

[23] Xiaowei Shang, Weiwei Jia, Jianchen Shan, Xiaoning Ding, and Cristian Borcea. Reestablishing Page Placement Mechanisms for Nested Virtualization. *IEEE Transactions on Cloud Computing*, 11(3):3239–3250, 2023.

[24] Standard Performance Evaluation Corporation. SPEC CPU2006 Benchmark. http://www.spec.org/cpu2006/.

[25] Stress-ng: a tool to load and stress a computer system. https://manpages.org/-stress-ng.

[26] George Taylor, Peter Davies, and Michael Farmwald. The TLB Slice - A Low-Cost High-Speed Address Translation Mechanism. In *Proceedings of the 17th annual international symposium on Computer Architecture*, 1990.

[27] Georgios Vavouliotis, Lluc Alvarez, Vasileios Karakostas, Konstantinos Nikas, Nectarios Koziris, Daniel A. Jiménez, and Marc Casas. Exploiting Page Table Locality

for Agile TLB Prefetching. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 85–98, 2021.

[28] Xiaodong Wang, Shuang Chen, Jeff Setter, and José F. Martínez. SWAP: Effective Fine-Grain Management of Shared Last-Level Caches with Minimum Hardware Support. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 121–132, 2017.

[29] Xiaolin Wang, Xiang Wen, Yechen Li, Yingwei Luo, Xiaoming Li, and Zhenlin Wang. A Dynamic Cache Partitioning Mechanism under Virtualization Environment. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1907–1911. IEEE, 2012.

[30] Zi Yan, Daniel Lustig, David Nellans, and Abhishek Bhattacharjee. Translation Ranger: Operating System Support for Contiguity-Aware TLBs. In *Proceedings of the 46th International Symposium on Computer Architecture*, 2019.

[31] [31] Ying Ye, Richard West, Zhuoqun Cheng, and Ye Li. COLORIS: A Dynamic Cache Partitioning System Using Page Coloring. In *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*, page 381-392, 2014.

[32] Heechul Yun, Renato Mancuso, Zheng-Pei Wu, and Rodolfo Pellizzoni. PALLOC: DRAM bank-aware memory allocator for performance isolation on multicore platforms. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 155–166. IEEE, 2014.

[33] Xiao Zhang, Sandhya Dwarkadas, and Kai Shen. Towards Practical Page Coloring-based Multi-core Cache Management. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 89–102, 2009.

RePort: Automatically mapping the attack surface of IoT Systems

Dimitrios Stefanos Porichis

ABSTRACT

The Internet of Things (IoT) is characterised by rapid, cost-focused development cycles, which have contributed to it becoming one of the most prominent cybersecurity concerns in today's technological landscape. One foundational step toward securing these systems could be mapping their attack surface. On that end, we introduce ***RePort:*** *An Automatic Attack Surface Mapper for firmware*.

We evaluate RePort on 10 firmware images from different device types and vendors, demonstrating its mapping capabilities by successfully identifying over 95 open ports, 63 binaries with access to external traffic, and 403 instances of known vulnerabilities (CVEs).

Keywords
Attack Surface, IoT, Firmware, Dynamic Analysis, Emulation, Automatic Mapping

ADVISOR
Thanassis Avgerinos, Assistant Professor, NKUA

1 INTRODUCTION

The technological landscape has experienced an unprecedented rise in the use of the Internet of Things (IoT), with 2024 estimates showing more than 18 billion IoT devices online [1]. Due to their rapid development-to-market cycles and aggressive cost-cutting strategies, manufacturers often treat security as a

secondary concern, sometimes even neglecting it entirely, with studies classifying 50% of devices as vulnerable to medium-to-high severity attacks [2,3,6].

Tackling the threat requires extensive manual effort and expertise, combined with the high costs of working with hardware. To overcome these barriers, researchers developed firmware-based solutions as a more scalable approach, developing both static and dynamic analysis tools [3].

Building on the progress made by the emulation community in recent years, we introduce ***RePort****: An Automatic Attack Surface Mapper for IoT Systems*. RePort incorporates both dynamic and static analysis, using actual runtimes to collect its target's behaviour and deriving conclusions from static logic parsers and Software Bill of Materials (SBOM) scanners [3, 7]. Similar systems provide solutions for identifying and mapping exposed services in web applications and online infrastructures [8]. However, none target firmware or embedded systems, nor do they operate within emulated runtime environments. In fact, ***RePort is the first system designed to perform attack surface mappings unassisted on firmware***.

Overall, our contributions consist of:

1. **Automatic attack surface mappings for IoT firmware.** We introduce RePort as a modular system that emulates Firmware and analyses system call traces to produce a stateful view of all exposed network ports, the binaries responsible for them, and their associated CVEs. [Section 3]
2. **General-purpose model for port activity tracking.** We propose the PortActivity class as a practical approach to deriving port mappings from low-level kernel logs, decoupled from output format. By incrementally tracking port-related system calls, it accurately reconstructs ownership, inheritance, and lifecycle of every network-facing socket observed during emulation, enabling precise attribution of network behaviour to individual binaries. [Subsection 3.3]
3. **An extended fork of FirmAE with enhanced system call tracking.** We develop a custom fork of FirmAE equipped with new kernels that expose richer system call probes, including return code extraction for port-

dependent calls, true port resolution for zero-port binds, and extended IPv6 tracking. [Subsection 3.4]

2 BACKGROUND AND RELATED WORK

Attack surface definitions vary based on perspective and have multiple semantic interpretations [9]. We define the attack surface based on the adversary viewpoint and, thus, its mapping. For the rest of this work, ***mapping the attack surface of a system*** *refers to the process of documenting all points where an unauthorised user could attempt to enter or extract data from a system*.

Assessing possible entrances is common in the industry, enabling organisations to identify potential vulnerabilities and effectively organise their security measures [10]. Deciding which components are significant for the mapping process usually depends on the product analysed and its dependencies.

IoT devices typically rely on network communication and offer minimal physical controls. Since their configuration and functionality are generally managed online, their primary points of entry, and thus their attack surface, are centred around network interfaces. Having the ability to monitor what ports are opened and by whom in that kind of firmware allows analysts to identify vulnerable, outdated, or unintended services that bad actors may exploit.

The absence of source code, however, primarily constrains the security posture evaluation of IoT devices. In fact, it is common that even the companies themselves lack access to said code, as they often acquire their hardware preloaded from OEM vendors [3]. Given these conditions, IoT security analysis is only possible at the binary level, making standard code analysis practices incompatible.

In our literature review, we found that the research community has two types of analysis tools, static and dynamic; however, effectively addressing IoT particularities remains an active research field. Static analysis examines firmware binaries [11, 12, 13] symbolically without executing them, identifying possible

vulnerabilities through dataflow and matching algorithms, often producing overwhelming false-positive rates.

Dynamic analysis, on the other hand, is based on emulating the given firmware, entirely or partially, and examining the services running by checking for known vulnerabilities, performing fuzzing, symbolic execution or using other custom tools. FirmAE [4] and its predecessor Firmadyne [14] were among the most pivotal dynamic analysis projects. Their frameworks posed the first attempts towards automated, scalable emulation of Linux-based embedded firmware images at a system level.

Interestingly, these frameworks generate a substantial amount of logs and artifacts, including filesystem dumps, system call tracking logs, Kernel information and Boot Loading sequences. This information allows for more in-depth analysis, such as OS-level behaviour tracing and Software Bill of Materials (SBOM) scanning, the practice of inventorying all components present in a software build [15]. SBOM can also correlate observed binaries with entries in public vulnerability databases, such as the Common Vulnerabilities and Exposures (CVEs) list [16].\

3 REPORT

Processing and contextualising the heterogeneous information produced by said tools manually requires significant time and expertise. As a response, we developed RePort, an automated framework designed to transform low-level system information into concise, stateful and externally verifiable attack surface mappings.

3.1 System Requirements

When working with IoT firmware, the mapping system should require as little information as possible, as source code is rarely available. We designed RePort with this in mind, operating entirely on black-box or grey-box inputs, without ever requiring access to source code.

- Given only a running device, it operates in **Blackbox** mode, using network scanning tools (e.g. nmap [5]) to enumerate open ports and summarise exposed services.
- When the firmware image itself is available, **Graybox** mode goes further; booting the image inside an emulation engine to produce a richer map that correlates each listening port with the responsible binary and any associated CVEs.

3.2 Architecture

To deliver this functionality, RePort employs a modular design that orchestrates various analysis tools. Figure 1 shows a top-down overview of RePort, showing the two main modes of operation.

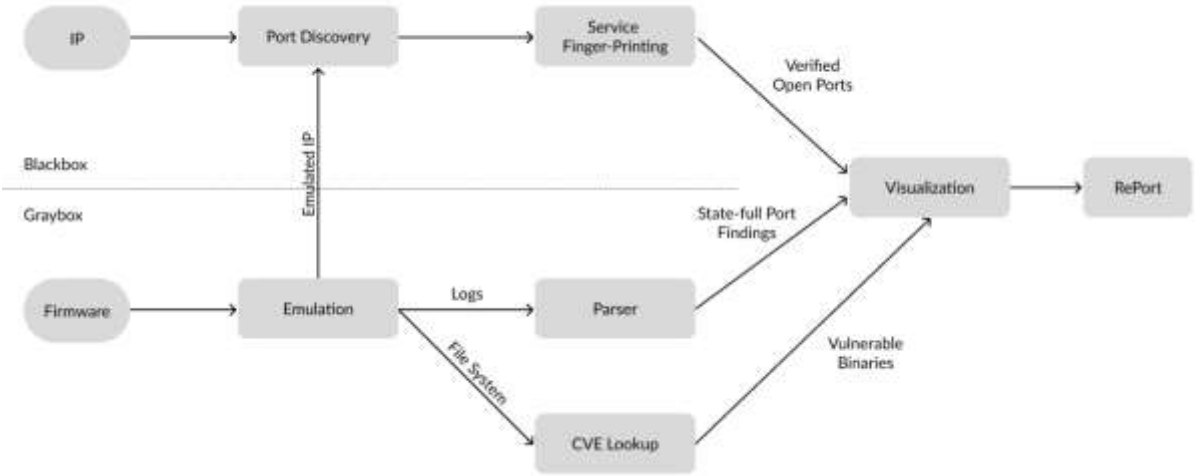


Figure 1: RePort's System Architecture

Blackbox mode is located at the top, taking an IP as input and performing Port Discovery and Service Fingerprinting on the given domain, producing a dictionary with all port findings.

Graybox mode is at the bottom and utilises firmware emulation to perform three distinct analyses on the target:

- Using a custom logic parser, **system call analysis** derives the binding, executing, and inheriting process relations, producing a dictionary of all operations performed in each port and by which process.
- Then, **known vulnerability identification** is performed on all files found inside the firmware's file system through the *CVE Lookup module*, utilising known vulnerability databases and SBOM, returning a map of all CVEs found on each binary.
- Finally, ***port confirmation*** *is performed on the* emulated target, attempting to ping the services identified by the system call analysis for added soundness.

All information is consolidated in the visualisation stage, where all data is condensed into a detailed Attack Surface Mapping overview - a RePort.

Firmware emulation, network mapping and SBOM analysis remain active and rapidly evolving areas of research. Acknowledging the field's dynamic nature, we designed ***all modules of RePort to be platform-independent***. At their core lie lightweight, abstract interfaces that allow integration with a wide range of sub-tooling. In the spirit of open science, RePort is open-source and available on its public GitHub Repository [18].

3.3 Emulation and System Call Analysis

In particular, we designed the emulation module to be compatible with diverse emulation platforms, provided they support system call tracking. We refer to this category of compatible tools as *Emulation Engines*.

Ultimately, a developer integrating an Emulation Engine with RePort needs only to hook up the necessary log files and integrate the execution UI, in line with the given tool's architecture. For example, our custom FirmAE fork, despite having the complex file structure of its predecessor, was straightforward to set up, requiring only the database, run scripts and execution scratch logs to operate (see Figure 2). Thorough documentation of the abstract classes provided by RePort is available at the project's repository [18] or in Section 3 of the complete thesis [19].

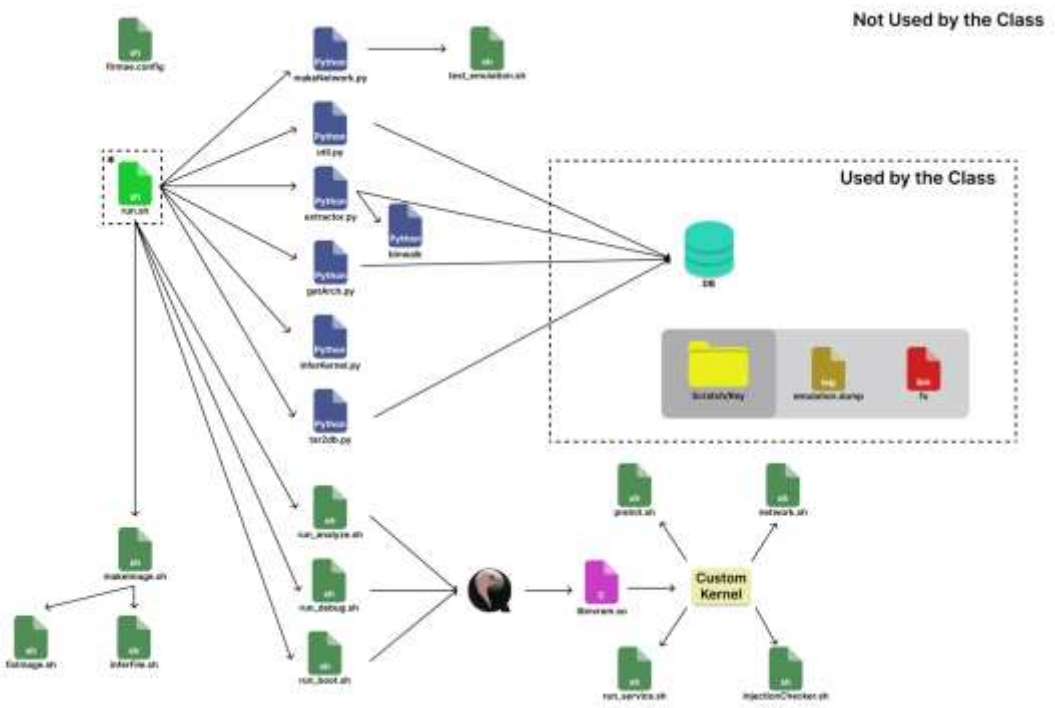


Figure 2: Custom FirmAE fork setup

Since different emulation tools produce distinct artifacts, system call output parsing varies per tool. Consequently, RePort does not offer a universal solution. Instead, it provides a general-purpose parsing class with dedicated methods for tracking all port-related system calls. These methods are intended to be invoked by format-specific parsers, separating port-tracking logic from the parsing concerns of each engine.

Based on these calls, PortActivity incrementally builds an internal model of port usage solely on the sequence of system calls observed during execution. It maintains stateful information about processes, file descriptors, ports, and protocol families, and it records both live and historical activity. Figure 3 depicts how PortActivity tracks the port changes in a simple bind and fork scenario.

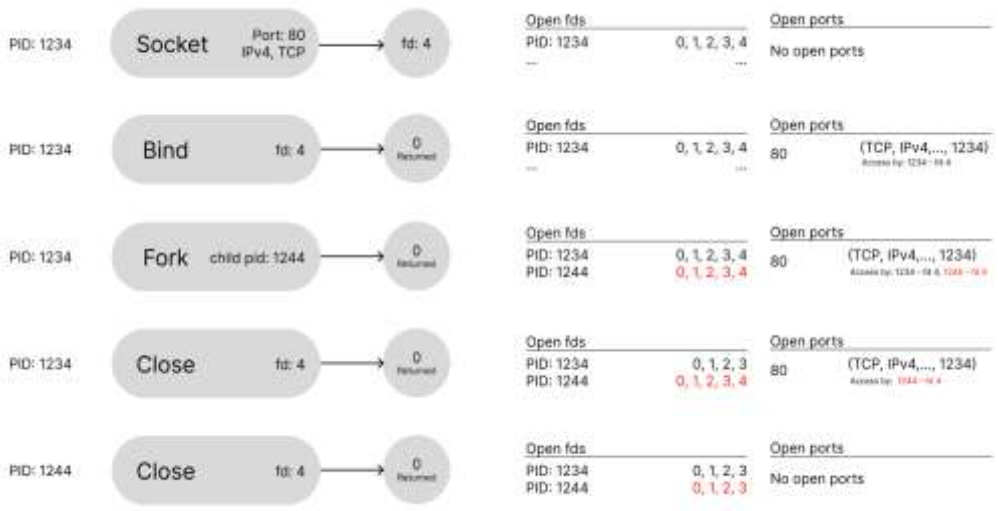


Figure 3: PortActivity tracking example

3.4 Custom FirmAE fork

RePort's mapping capabilities are fundamentally constrained by the Emulation Engine used, both in terms of the firmware it supports and the granularity of information that can be extracted from its artifacts. For our default emulation engine, we implemented a custom FirmAE fork with additional system call-tracking capabilities. Our fork still adheres to the same design principles as its predecessor, but with new custom kernels equipped with more verbose system call probes and updated dependencies for the deprecated packages. More precisely, our changes include:

- Extracting the return code of port-dependent system calls to check for failure.
- Retrieving the true port number assigned to a socket when bind is called with port zero as an argument.
- Extending tracking capabilities for the IPv6 protocol.
- Modifying emulation options for consistency.

Through these changes, FirmAE utilises all tracking capabilities of RePort, ultimately creating a complete mapping solution. Our fork is publicly available on GitHub under its own repository [20].

3.5 CVE Lookup and SBOM

RePort uses SBOM to identify known vulnerabilities through its CVE Lookup module, by scanning executables retrieved from the target filesystem. Emulation Engines with filesystem extraction capabilities may invoke CVE Lookup to analyse binaries and identify matches with known vulnerable versions.

The unified findings of CVE Lookup and the Parser module serve as a mapping of the firmware's attack surface. By correlating each executable's runtime network behaviour with its security posture, RePort enables analysts to pinpoint which components are both externally exposed and potentially exploitable. As a result, it provides a prioritised view of risk that guides remediation, hardening, or further manual auditing.

In the current version of RePort, CVE Lookup is powered by the open-source vulnerability scanner Grype [17], which was selected for its active development, strong community support, and robust handling of diverse packaging formats, including support for raw binaries.

3.6 Confirmation and Blackbox Analysis

The Confirmation operation enhances the soundness of the findings produced by RePort**,** by using the network discovery tool Nmap [5] to cross-check the services identified through system call parsing. Once a port is flagged as open or associated with a vulnerable binary, RePort optionally initiates a probe to determine whether the corresponding service is active and responsive in a second confirmation emulation. This approach provides an additional layer of confidence that the observed behaviour corresponds to a real reachable service, rather than a transient or prematurely exited one.

To further lower the analysis requirements of RePort, we abstracted this probing mechanism into a standalone capability, referred to as Blackbox Analysis. When

used independently, this feature enables external mapping of firmware behaviour by scanning the network environment for open ports and accessible services without relying on prior system call parsing.

Blackbox Analysis is especially useful when the firmware of a given IoT device is not available, but a runtime is, such as a smart wall plug on a local network. Thus, Blackbox Analysis complements the internal tracking performed by Graybox Analysis, or functions entirely on its own in different analysis scenarios.

3.7 Visualisation

Finally, RePort produces an HTML Report that presents the analysis results in a structured and accessible format. The report begins with an overview of port activity, organised by port number, providing a high-level view of active services and potential entry points into the system's attack surface (Figure 4). Ports confirmed to be open are explicitly highlighted due to their increased relevance in security assessments.

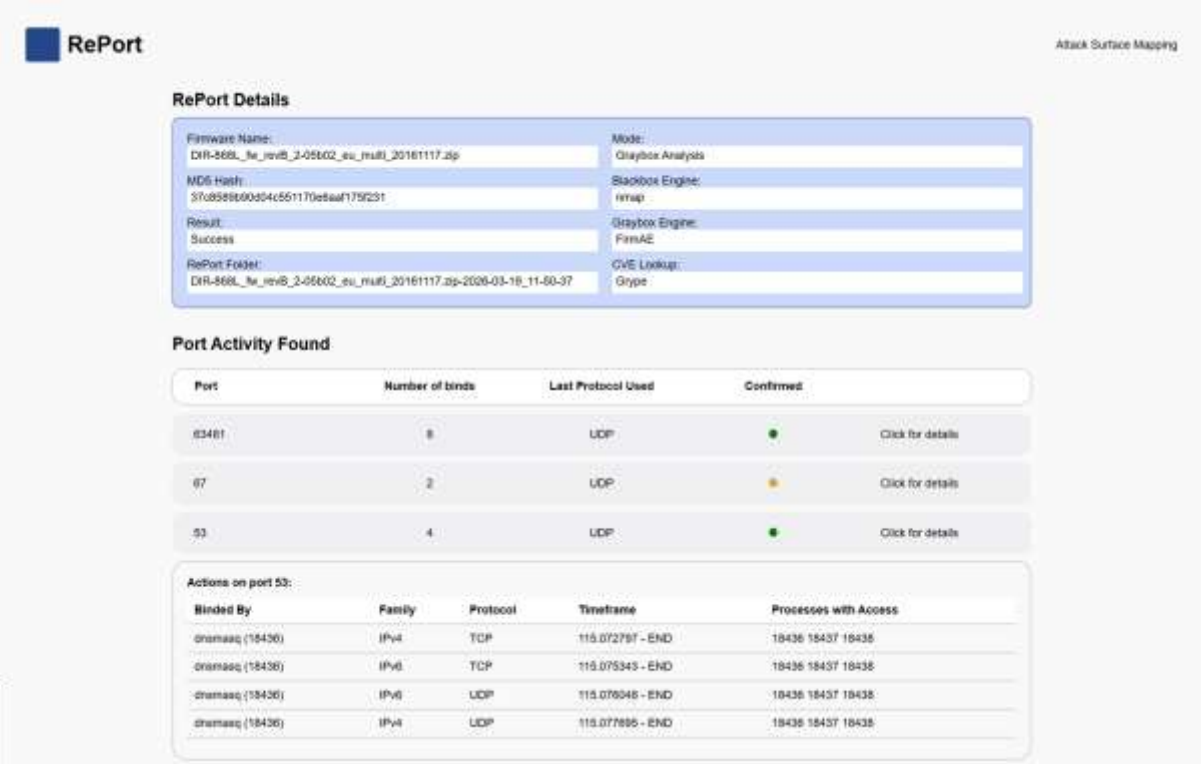


Figure 4: RePort Sample - Port activity

Subsequently, the report presents a per-binary analysis of system activity (Figure 5). For each binary, a minimal overview summarises key characteristics, followed by a detailed table listing all associated vulnerabilities (Figure 6).

Figure 5: RePort Sample - Outward facing binaries

RePort also stores a collection of artifacts for each analysis in a directory, including the extracted firmware file system, system call logs, CVE findings, and the generated RePort.html. These artifacts ensure traceability and enable subsequent re-examination or auditing of the analysis results.

Figure 6: RePort Sample - CVEs overview

4 EVALUATION

To evaluate RePort's performance, we conducted mapping experiments on 10 different firmware. Our dataset was composed of firmware images from routers, IP cameras, and network extenders manufactured by D-Link, TRENDnet, Netgear, TP-Link, ASUS, and Belkin. We randomly selected images from FirmAE's running dataset to ensure compatibility with emulation and are available for download through the Google Drive link listed in RePort's GitHub repository [18].

Table 1: Experimental Results

Firmware Details			Port Statistics		CVE Statistics		Time Statistics		
A/A	Type	Vendor	Ports Used	Instances	Critical Binaries	CVEs	Time	Firmware Size	Syscall.log & RePort size
1	IP Cam	DLink	20	32	10	69	9:39	~13MB	2.27MB
2	Router	DLink	20	28	11	17	9:33	~13MB	2.53MB
3	Router	Netgear	15	20	13	80	11:56	~30MB	477KB
4	Router	Netgear	13	17	9	17	6:34	~11.8MB	298KB
5	IP Cam	DLink	9	17	3	60	11:55	~46MB	655KB
6	Router	Asus	6	7	5	17	16:49	~1.8MB	76KB
7	Router	Belkin	5	5	5	17	6:03	~2.8MB	251KB
8	Router	Trendnet	3	3	4	17	5:56	~4.2MB	74.1KB
9	IP Cam	Trendnet	2	2	2	86	9:33	~6.6MB	160KB
10	Extender	TPLink	2	2	1	23	9:42	~4.7MB	35.7KB
Total			95	133	63	403	1:37:40	-	-

Through our experiments, RePort successfully produced attack surface mappings for all selected firmware images, identifying over 95 ports utilised by 133 instances. A total of 63 distinct binaries were flagged as critical, as they are capable of accepting external traffic. The CVE Lookup module, identified over 403 CVE appearances in 14 binaries found within the emulated firmware.

Remarkably, our manual analysis of some critical binaries revealed that this is only a lower bound of the actual number of CVEs. In a DLink Router (AA 2 in Table

1), for example, RePort successfully identifies a *dnsmasq* binary from 2016 having access to port 53. Although this binary has multiple CVEs, Grype was unable to identify the threat, highlighting that the available security tools are not complete.

Time-wise, the average execution time for our dataset15 was 9 minutes and 46 seconds, with some divergence attributed to firmware behaviour and emulation noise. Based on these time requirements, RePort is suitable for manufacturers' Continuous Integration pipelines (CI), producing an attack surface report for every developer push.

Regarding the output storage size, the average compressed RePort folder produced during our runs was approximately 1MB, excluding the decompressed file system; a size negligible compared to the actual firmware image size.

All experiments were conducted using GitHub Actions, with all execution results available on RePort's GitHub [18] repository as Action Artifacts.

5 CONCLUSIONS

We proposed RePort as a framework for leveraging state-of-the-art analysis tools to generate reports that enhance the security posture of IoT devices. Our experimental results demonstrate that RePort is able to extract critical metadata, identify exposed services, and highlight vulnerabilities across a range of firmware samples.

As available sub-tools continue to evolve, RePort's performance will follow, reducing false-negative rates and broadening firmware support. Its time and storage requirements allow integration with CI/CD pipelines and its modular and open-source design further enables new features to be introduced, such as fuzzing and additional output options.

REFERENCES

[1] IoT Analytics. State of IoT 2024: Number of connected IoT devices growing 13 https://iot-analytics.com/number-connected-iot-devices/.

[5] Unit 42. 2020 Unit 42 IoT Threat Report — unit42.paloaltonetworks.com. https://unit42.paloaltonetworks.com/iot-threat-report-2020/.

[6] Ibrahim Nadir, Haroon Mahmood, and Ghalib Asadullah. A taxonomy of iot firmware security and principal firmware analysis techniques. International Journal of Critical Infrastructure Protection, 38:100552, 2022.

[7] [Mingeun Kim, Dongkwan Kim, Eunsoo Kim, Suryeon Kim, Yeongjin Jang, and Yongdae Kim. Firmae: Towards large-scale emulation of iot firmware for dynamic analysis. In Proceedings of the 36th Annual Computer Security Applications Conference, ACSAC '20, page 733–745, New York, NY, USA, 2020. Association for Computing Machinery.

[8] Nmap. Nmap: the Network Mapper - Free Security Scanner — nmap.org. https://nmap.org/.

[9] Eryk Schiller, Andy Aidoo, Jara Fuhrer, Jonathan Stahl, Michael Ziörjen, and Burkhard Stiller. Landscape of iot security. Computer Science Review, 44:100467, 2022.

[10] Xiaotao Feng, Xiaogang Zhu, Qing-Long Han, Wei Zhou, Sheng Wen, and Yang Xiang. Detecting vulnerability on iot device firmware: A survey. IEEE/CAA Journal of Automatica Sinica, 10(1):25–41, 2023.

[11] Andreas Georgiou. GitHub - superhedgy/AttackSurfaceMapper: AttackSurfaceMapper is a tool that aims to automate the reconnaissance process. — github.com. https://github.com/superhedgy/AttackSurfaceMapper.

[12] Christopher Theisen, Nuthan Munaiah, Mahran Al-Zyoud, Jeffrey C Carver, Andrew Meneely, and Laurie Williams. Attack surface definitions: A systematic literature review. Information and Software Technology, 104:94–103, 2018.

[13] Mengyuan Zhang, Lingyu Wang, Sushil Jajodia, and Anoop Singhal. Network attack surface: Lifting the concept of attack surface to the network

level for evaluating networks' resilience against zero-day attacks. IEEE Transactions on Dependable and Secure Computing, 18(1):310–324, 2021.

[14] Yaniv David, Nimrod Partush, and Eran Yahav. Firmup: Precise static detection of common vulnerabilities in firmware. SIGPLAN Not., 53(2):392–404, March 2018.

[15] Zicong Gao, Chao Zhang, Hangtian Liu, Wenhou Sun, Zhizhuo Tang, Liehui Jiang, Jianjun Chen, and Yong Xie. Faster and better: Detecting vulnerabilities in linux-based iot firmware with optimized reaching definition analysis. In NDSS, 2024.

[16] Wil Gibbs, Arvind S Raj, Jayakrishna Menon Vadayath, Hui Jun Tay, Justin Miller, Akshay Ajayan, Zion Leonahenahe Basque, Audrey Dutcher, Fangzhou Dong, Xavier Maso, Giovanni Vigna, Christopher Kruegel, Adam Doupé, Yan Shoshitaishvili, and Ruoyu Wang. Operation mango: scalable discovery of taint-style vulnerabilities in binary firmware services. In Proceedings of the 33rd USENIX Conference on Security Symposium, SEC '24, USA, 2024. USENIX Association

[17] Daming D Chen, Maverick Woo, David Brumley, and Manuel Egele. Towards automated dynamic analysis for linux-based embedded firmware. In NDSS, volume 1, pages 1–1, 2016.

[18] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. Software bills of materials are required. are we there yet? IEEE Security & Privacy, 21(2):82–88, 2023.

[19] Common Vulnerabilities. Common vulnerabilities and exposures. The MITRE Corporation,[online] Available: https://cve. mitre. org/index. html, 2005.

[20] Grype. GitHub - anchore/grype: A vulnerability scanner for container images and filesystems — github.com. https://github.com/anchore/grype.

[21] Dimitris Porichis. GitHub - DPorichis/RePort: RePort - Automatically mapping the attack surface of IoT firmware using Dynamic Analysis — github.com. https://github.com/DPorichis/RePort.

[22] Porichis Dimitrios-Stefanos. RePort: Automatically Mapping the Attack Surface of IoT Systems. year = "2025", School of Science, Department of Informatics and Telecommunications, National and Kapodistrian University of Athens. https://pergamos.lib.uoa.gr/en/item/uoadl:5298750

[23] George-Rg. (n.d.). GitHub - George-RG/FirmaInc: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis. GitHub. https://github.com/George-RG/FirmaInc.

Contrastive Unlearning: Selective Class Forgetting by Preserving Optimality Conditions in the Representation Space

Eleni Fili

ABSTRACT

Self-supervised contrastive learning (CL) has emerged as a dominant paradigm in representation learning, enabling powerful models to be trained on large-scale unlabeled datasets. However, its widespread deployment raises privacy and compliance concerns, as CL models may retain sensitive information about individual training samples, conflicting with regulations such as the GDPR's right to be forgotten. While machine unlearning offers a computationally efficient alternative to full retraining, existing approaches are limited in two important ways: they are primarily developed for supervised, label-dependent settings and thus are poorly suited to the geometry-driven objectives of contrastive learning, and they are typically evaluated through downstream task performance alone, which may overlook residual information in representation space. To address these gaps, we propose a principled framework for class-wise unlearning in self-supervised contrastive learning. Our method combines a contrastive retention objective that preserves the optimality conditions of retained representations with a proxy-guided forgetting objective that aligns forgotten samples to a blindspot model trained only on retained data. We further reinforce forgetting through attention transfer to suppress residual class-specific activations. Beyond downstream evaluation, we introduce a comprehensive assessment protocol based on representation-level metrics and membership inference analysis. Experiments on CIFAR-10 show that our method reduces forgotten-class accuracy to 18%, preserves 82% accuracy on retained classes, and achieves near-

random membership inference attack success (0.54), while closely approximating full retraining at substantially lower computational cost.

ADVISOR

Yannis Panagakis, Associate Professor, NKUA

1 INTRODUCTION

The emergence of self-supervised learning (SSL) has fundamentally transformed modern machine learning by enabling models to learn rich and generalizable representations from large-scale unlabeled data, alleviating the reliance on costly human annotation [1,2]. Among SSL paradigms, contrastive learning (CL) has emerged as a dominant approach for representation learning. By encouraging agreement between positive pairs while repelling negative pairs in the embedding space, contrastive methods achieve state-of-the-art performance across a wide range of tasks [1,3]. Their effectiveness is commonly attributed to two complementary properties: alignment of positive pairs and uniformity of the representation distribution on the unit hypersphere [4].

However, the widespread deployment of SSL models trained on internet-scale data raises critical privacy and regulatory concerns [5]. These models can memorize and potentially leak sensitive information about training samples [6], creating significant legal and ethical challenges. Regulatory frameworks such as the European Union's General Data Protection Regulation (GDPR) have formalized these concerns through provisions like the "right to be forgotten" (Article 17) [7], which grants individuals the legal authority to request deletion of their personal data.

The straightforward solution—retraining from scratch after removing the target data—is computationally prohibitive for modern large-scale models [8] as **(1)** it requires enormous computational resources equivalent to the original training

cost, **(2)** creates unacceptable delays in legal compliance, and **(3)** becomes increasingly infeasible as model and dataset sizes continue to grow exponentially. These fundamental limitations have motivated the development of machine unlearning [9]; efficient algorithms to approximate the retrained model by selectively removing data influence while preserving model utility.

Limitations of existing works. Despite growing interest, existing unlearning approaches suffer from two key limitations that constrain their applicability to modern SSL systems. The first is the **SSL gap**: most machine unlearning methods have been studied primarily in supervised classification settings, often centered on label-driven objectives and classifier behavior [9,10]. In contrast, SSL—especially contrastive learning—optimizes relational, geometry-driven objectives in embedding space, making supervised unlearning strategies poorly suited to this setting. The second is **evaluation inadequacy**: current protocols assess unlearning mainly through downstream task performance [9,11], which fails to reveal residual information preserved in representation geometry. Such latent traces may remain exploitable by adversaries, undermining the reliability of unlearning guarantees.

Our approach. To address these challenges, we propose the first principled unlearning framework designed specifically for contrastive learning systems. The core challenge lies in *selective disruption*: eliminating the geometric influence of forgotten samples while preserving the structural properties that underpin useful representations. To this end, we utilize a blindspot model framework [12], in which a lightweight proxy model trained solely on the retain set serves as a counterfactual reference for the ideal state where the forgotten data had never shaped the learned representation. Building on this idea, we formulate contrastive unlearning as a dual-objective optimization problem: a retention objective applied to retained samples to preserve alignment and uniformity, and a forgetting objective applied to forgotten samples to align their embeddings with blindspot counterparts while using attention transfer penalties to weaken class-specific feature associations [13]. Finally, to address the limitations of performance-based evaluation, we develop a comprehensive suite of metrics grounded in neural collapse theory and contrastive optimality. By measuring

alignment, uniformity, and effective rank, our framework evaluates unlearning efficacy directly in embedding space.

2 BACKGROUND AND RELATED WORK

2.1 Self-Supervised Contrastive Learning

Self-supervised learning (SSL) aims to learn transferable representations without external annotations by constructing supervisory signals directly from the data. In vision, contrastive learning has emerged as one of the most effective paradigms: two augmented views of the same image are treated as a positive pair, while views from different images are treated as negatives. Methods such as SimCLR and MoCo demonstrated that this formulation can produce representations competitive with supervised pretraining across diverse downstream tasks [1,3]. Beyond negative-pair methods, non-contrastive and redundancy-reduction approaches, such as Barlow Twins and VICReg, showed that representation collapse can also be prevented through explicit regularization of feature statistics [14,15]. Earlier SSL paradigms such as DeepCluster, further support the broader perspective that representation quality depends on the structure induced in feature space rather than external labels [2].

For an encoder f, contrastive learning (CL) is commonly implemented via an InfoNCE-style objective [16], which encourages representations of positive pairs to be similar while repelling them from other samples in the minibatch [1,16]. Although the precise formulation varies across methods, the shared objective is to learn embeddings that are both invariant to augmentations and sufficiently discriminative across samples [1,4]. In the setting of machine unlearning for CL, this is particularly important because unlearning is applied not only to a linear head, but to a pretrained representation model whose geometry already captures semantic structure.

2.2 Geometry of Contrastive Representations

A useful theoretical lens for contrastive learning is that it implicitly optimizes two geometric objectives: alignment, which encourages representations of positive pairs to be close, and uniformity, which promotes an even distribution on the unit hypersphere. Wang and Isola formalized this perspective, showing that high-quality contrastive representations emerge from jointly optimizing these properties [4]. This view further motivates evaluating unlearning beyond target-class accuracy, by examining how the geometry of the learned representation space is modified after forgetting.

Subsequent theoretical work has further linked contrastive objectives to structured optima in representation space. In particular, for InfoNCE-family objectives, optimal solutions can exhibit perfect alignment together with regular-simplex separation in the mini-batch regime, while asymptotically recovering the alignment-uniformity characterization on the sphere [17]. These results suggest that alignment and global separation are intrinsic properties of contrastive optima, not just incidental by-products of the training dynamics [4,17]. This observation is central to our setting: selective forgetting should eliminate information about the target class while preserving, as much as possible, the geometric optimality conditions that govern the representations of the retained data.

2.3 Machine Unlearning

Machine unlearning studies how to remove the influence of a subset of training data from a trained model without retraining from scratch. Early work by Cao and Yang introduced the problem of enabling learning systems to forget efficiently [18] and subsequent work formalized notions of deletion efficiency and practical removal procedures for specific models such as k-means [19]. For deep networks, full retraining on the retained dataset remains the gold standard for exact deletion [19]. However, this approach is often computationally prohibitive, especially when deletion requests are frequent or large in scale [19]. This has motivated a wide range of approximate unlearning methods that trade exactness for efficiency [19-22]. Recent surveys classify machine unlearning methods into

exact, approximate, and certified or verified approaches, while also covering application-specific settings and use cases [8-10].

A major line of research focuses on deep model unlearning through parameter-space updates designed to approximate the effect of retraining. Representative methods include gradient-based removal techniques such as SalUn [21] and amnesiac-style updates [20], which attempt to remove the contribution of the forget set while preserving performance on the retained data. Another direction modifies the training procedure itself to facilitate future deletion requests, for example through partitioning-based strategies such as SISA [22] that enable selective retraining of model components. While these approaches can be effective in certain regimes, they largely operate in parameter space and do not leverage the geometric structure of the learned representation.

For our setting, we focus on class-wise unlearning, where all samples belonging to a target class must be removed. Recent approaches such as Boundary Unlearning and Deep Unlearning address this challenge by modifying, respectively, the classifier's decision boundary and class-discriminative internal representations, while achieving substantial speedups over retraining [23,24]. However, these methods are still primarily driven by output-space behavior.

Motivated by the need for stronger guarantees, recent work has begun to study certified unlearning, which asks whether a model after deletion is statistically indistinguishable from one that never saw the deleted data. Guo et al. make this precise for linear models [25], while Sekhari et al. develop broader algorithmic foundations for machine unlearning [26]. These notions are conceptually related to differential privacy, since both reason about output stability under neighboring datasets [27]. In practice, though, strong certification remains out of reach for modern deep neural networks, particularly in large-scale vision settings [27]. Accordingly, recent literature has shifted toward approximate unlearning methods supported primarily by empirical evaluation rather than exact guarantees [8-10,27].

3 METHODOLOGY

3.1 Problem Setup

Let $\mathcal{D}$ denote the training dataset partitioned into a retain set $\mathcal{D}_r$ and a forget set $\mathcal{D}_f$, such that

$$\mathcal{D} = \mathcal{D}_r \cup \mathcal{D}_f, \qquad \mathcal{D}_r \cap \mathcal{D}_f = \emptyset.$$

The goal of machine unlearning is to update a pretrained model $\mathcal{M}_\theta$ so that information about $\mathcal{D}_f$ is removed while preserving performance on $\mathcal{D}_r$.

We formulate this objective as a constrained optimization problem:

$$\min_{\theta} \mathcal{R}(\boldsymbol{\theta}; \mathcal{D}_r) \quad \text{s.t.} \quad \mathcal{F}\big(\boldsymbol{\theta}; \mathcal{D}_f\big) \geq \lambda,$$

where $\mathcal{R}$ quantifies retained utility, $\mathcal{F}$ measures forgetting strength and $\lambda > 0$ specifies the desired minimum forgetting threshold.

To allow more flexibility during optimization, the constraint can be relaxed and incorporated into the objective as a penalty. This leads to the following unconstrained formulation:

$$\min_{\theta} \mathcal{R}(\boldsymbol{\theta}; \mathcal{D}_r) - \beta \cdot \mathcal{F}(\boldsymbol{\theta}; \mathcal{D}_f),$$

where $\beta > 0$ is a hyperparameter that controls the retention-forgetting trade-off.

3.2 Kernel Contrastive Retention

To preserve the geometric structure of the latent space for samples belonging in the retain set—specifically, to ensure continued optimization toward geometric optimality conditions—we adopt the Kernel Contrastive Loss (KCL) framework [17]. KCL balances the alignment of corresponding representations with the dispersion of mismatched pairs in a learned feature space.

Formally, given two sets of representations $\mathbf{U} = \{\mathbf{u}_i \in \mathbb{R}^d\}_{i=1}^M$ and $\mathbf{V} = \{\mathbf{v}_i \in \mathbb{R}^d\}_{i=1}^M$,

representing two views of the same batch of data instances, the Kernel Contrastive Loss is defined as:

$$\mathcal{L}_{KCL}(\mathbf{U},\mathbf{V}) = -\frac{1}{M}\sum_{i=1}^{M} K_A(\mathbf{u}_i,\mathbf{v}_i) + \frac{\gamma}{M(M-1)}\sum_{\substack{i,j=1 \\ j\neq i}}^{M} K_U(\mathbf{u}_i,\mathbf{v}_i),$$

The first term encourages high similarity between matching pairs by maximizing the alignment kernel $K_A(\cdot,\cdot)$. The second term introduces a repulsive force by penalizing similarities between non-corresponding pairs $(\mathbf{u}_i,\mathbf{v}_j)$ for $i \neq j$, using the uniformity kernel $K_U(\cdot,\cdot)$. The scaling factor $\gamma > 0$ controls the trade-off between attraction and repulsion, while normalization by $M(M-1)$ accounts for the number of unique pairwise comparisons.

In our implementation, we employ the Gaussian radial basis function (RBF) kernel for both alignment and uniformity components. The RBF kernel is chosen due to its smoothness, differentiability, and sensitivity to local variations in the embedding space. Its exponential decay with respect to distance makes it especially effective for distinguishing between positive and negative samples in self-supervised settings.

The retention objective is applied exclusively to samples from the retain set within each training batch. This ensures that the semantic structure of the retained data is preserved, while preventing interference from samples belonging to the forget set. The resulting retain loss is given by:

$$\mathcal{L}_r = \mathcal{L}_{KCL}(\mathbf{U}_r,\mathbf{V}_r),$$

where $\mathbf{U}_r$ and $\mathbf{V}_r$ denote the feature embeddings corresponding to retain-set samples.

3.3 Proxy-Guided Forgetting

To enforce the removal of information pertaining to $\mathcal{D}_f$, we adopt a proxy-based strategy inspired by [23]. The core idea is to guide the unlearning process using a reference—referred to as the blindspot model $\mathcal{B}$—which approximates the behavior of a network trained from scratch solely on the retain set.

During unlearning, we encourage the original pretrained model $\mathcal{M}_\theta$ to mimic the behavior of this blindspot model specifically on the forget set. The blindspot model is initialized from scratch and trained exclusively on $\mathcal{D}_r$ for a small number

of epochs. This setup ensures that $\mathcal{B}$ is inherently "blind" to the forgotten data, making it a reliable target for inducing controlled forgetting in $\mathcal{M}_\theta$.

The goal of unlearning in this context is to counteract the structured alignment and uniform dispersion that the forgotten samples contributed during original training. By applying the Kernel Contrastive Loss (KCL) between the current model's representations and those of the proxy, we implicitly force the forget-set embeddings to collapse toward the proxy's geometry, one that lacks any class-discriminative imprint from the forgotten data. This weakens both intra-class compactness and inter-class separation, thereby breaking the geometric optimality conditions that would otherwise persist.

Specifically, for a batch containing k forget samples, we extract the anchor representations $\mathbf{M}_f = \{\mathbf{m}_i \in \mathbb{R}^d\}_{i=1}^k$ using $\mathcal{M}_\theta$ and the positive representations $\mathbf{B}_f = \{\mathbf{b}_i \in \mathbb{R}^d\}_{i=1}^k$ using $\mathcal{B}$. These are passed through the KCL, encouraging $\mathcal{M}_\theta$ to align its latent space with $\mathcal{B}$ for the forget samples:

$$\mathcal{L}_{KCL}^{forget} = \mathcal{L}_{KCL}(\mathbf{M}_f, \mathbf{B}_f),$$

This alignment term promotes similarity between corresponding pairs $(\mathbf{m}_i, \mathbf{b}_i)$ while preserving contrastive separation across non-matching pairs $(\mathbf{m}_i, \mathbf{b}_j)$ for $i \neq j$, effectively reshaping the forget space without causing representational collapse.

Attention Transfer Regularization. Representation alignment alone may not sufficiently suppress internal feature activations associated with forgotten classes; to address this, we additionally regularize the model's internal attention patterns. We adopt a variant of the Attention Transfer (AT) loss, typically used in knowledge distillation [13], and repurpose it for unlearning. This loss compares the spatial activation maps produced by the projection heads of $\mathcal{M}_\theta$ and $\mathcal{B}$ on forget samples, penalizing discrepancies between their attention distributions.

For a batch of k forget samples, let $\mathbf{M}_f^{(\ell)} \in \mathbb{R}^{k\times d}$ and $\mathbf{B}_f^{(\ell)} \in \mathbb{R}^{k\times d}$ be the layer $\boldsymbol{\ell}$ activation tensors of the model and the proxy, respectively. The attention map for a given sample is computed by squaring the activations, averaging across channels, flattening spatial dimensions and applying ℓ_2 normalization. The AT loss is defined as:

$$\mathcal{L}_{AT} = \frac{1}{L}\sum_{\ell=1}^{L}\left\|\mathrm{att}\left(\mathbf{M}_f^{(\ell)}\right) - \mathrm{att}\left(\mathbf{B}_f^{(\ell)}\right)\right\|^2.$$

This term promotes alignment between the saliency profiles of the two models, actively discouraging the unlearned model from attending to spatial regions that are predictive of the forgotten information.

We define the forget loss as a composite of the proxy-guided representation alignment and activation-based attention transfer losses:

$$\mathcal{L}_f = \alpha \cdot \mathcal{L}_{KCL}^{forget} + \beta \cdot \mathcal{L}_{AT},$$

where α and β are hyperparameters controlling the contribution of each term.

3.4 Training Objective

Following the relaxed optimization formulation introduced in section 3.1, we define our unlearning objective as a weighted combination of the retain loss $\mathcal{L}_r$ and the forget loss $\mathcal{L}_f$. Specifically, the total loss used during unlearning is formulated as:

$$\mathcal{L} = \mu \cdot \mathcal{L}_f + (1-\mu) \cdot \mathcal{L}_r,$$

where the dynamic weighting coefficient $\mu = \frac{|\mathcal{D}_f \cap \mathrm{batch}|}{|\mathrm{batch}|}$ represents the proportion of forget samples in the current batch. This adaptive weighting ensures that the gradient updates are proportionally scaled based on the empirical distribution of the batch, maintaining strict equilibrium between utility preservation and information erasure.

4 EXPERIMENTS

4.1 Experimental setup

Unlearning protocol. We study class-wise machine unlearning on CIFAR-10. In each experiment, one class is designated as the forget set $\mathcal{D}_f$, while the remaining

nine classes form the retain set $\mathcal{D}_r$. This yields ten class-removal tasks, allowing us to evaluate both the effectiveness of forgetting and the consistency of each method across semantic categories.

Representation model. Our encoder follows a SimCLR-style architecture consisting of a CIFAR-adapted ResNet-18 backbone and a two-layer MLP projection head. The encoder produces 512-dimensional representations, which are projected into a 128-dimensional latent space where the contrastive objective is optimized. To assess representation quality, we adopt the standard linear evaluation protocol: after pretraining or unlearning, the encoder is frozen and a linear classifier is trained on top of the 512-dimensional features.

Baselines. We compare our proposed method against three baselines. ***Retrain*** trains a new model from scratch using only $\mathcal{D}_r$ and serves as the reference unlearning target. ***Gradient Ascent*** performs ascent updates on $\mathcal{D}_f$, explicitly driving the model away from the forgotten samples. ***Fine-Tuning*** continues optimization only on $\mathcal{D}_r$, relying on catastrophic forgetting to remove the influence of the forgotten class.

Pretraining and linear evaluation. Unless otherwise stated, pretraining is performed for 200 epochs using SGD with momentum 0.9, batch size 256, and cosine learning-rate decay. We optimize KCL with a Gaussian kernel using $t = 2.0$ and $\gamma = 20$. Data augmentation follows the standard SimCLR pipeline. For linear evaluation, we train a single fully connected classifier on frozen encoder features for 50 epochs using SGD. The retraining baseline uses the same configuration, with training restricted to $\mathcal{D}_r$.

Unlearning configuration. For our proposed method, the pretrained encoder is further optimized for 10 epochs. The forgetting objective combines the KCL-based forgetting term with the attention transfer loss, weighted by $\alpha = 10$ and $\beta = 50$, respectively. The proxy model is trained from scratch on $\mathcal{D}_r$ for 20 epochs. For the ***Gradient Ascent*** baseline, we perform 50 epochs with learning rate 5×10^{-4}, which provided the most stable behaviour in preliminary experiments. Finally, ***Fine-Tuning*** is applied on $\mathcal{D}_r$ for 20 epochs.

Evaluation metrics. To ensure a controlled comparison across methods, we keep the evaluation protocol fixed and attribute performance differences solely to changes in the encoder. We report **retained-class accuracy** on both the training and test partitions $(\mathrm{Train}_r, \mathrm{Test}_r)$, **forgotten-class accuracy** on the corresponding partitions $(\mathrm{Train}_f, \mathrm{Test}_f)$, and the **Attack Success Rate** (ASR) of a membership inference attack. In addition, we analyze representation geometry using **Alignment**, **Uniformity**, **Inter-Class Effective Rank** on retained classes, and **Intra-Class Effective Rank** on forgotten classes.

4.2 Downstream Performance and Privacy Evaluation

To evaluate unlearning efficacy, we first assess utility preservation and forgetting effectiveness, as shown in **Table 1**. We observe that our proposed method performs competitively in utility preservation, scoring approximately 0.84 on the retained train set and 0.82 on the test set. The slight degradation of roughly 5% compared to the retrained baseline reflects the residual influence, which is expected given that our method updates a pretrained model to remove the contribution of specific data points. Concurrently, our method demonstrates strong forgetting capabilities, lowering the average accuracy of the forgotten class to 0.19 on the train set and 0.18 on the test set. While Fine-Tuning yields strong retention scores, it critically fails to erase the influence of the forget class, maintaining high accuracies. Conversely, Gradient Ascent demonstrates the lowest utility retention, which likely stems from adversarial nature of the updates, which can interfere with shared features across both retained and forgotten classes, while achieving moderate forgetting.

Table 1. Retention, forgetting, and memorization metrics across different unlearning methods. Values in bold indicate the best performance.

	Retrain					**Our Method**				
Class	Train_r	Train_f	Test_r	Test_f	ASR	Train_r	Train_f	Test_r	Test_f	ASR
0	0.89	0.00	0.87	0.00	0.53	0.84	0.39	0.82	0.36	**0.54**
1	0.88	0.00	0.86	0.00	0.62	0.82	**0.07**	0.81	**0.06**	0.54
2	0.91	0.00	0.89	0.00	0.58	0.85	**0.21**	0.84	**0.21**	**0.54**

3	0.92	0.00	0.90	0.00	0.65	0.87	**0.15**	0.86	**0.14**	0.54
4	0.90	0.00	0.88	0.00	0.63	0.85	**0.12**	0.83	**0.10**	0.53
	Gradient Ascent					**Fine-Tuning**				
0	0.82	**0.33**	0.79	**0.31**	0.58	**0.88**	0.63	**0.86**	0.60	0.50
1	0.80	0.39	0.79	0.38	**0.63**	**0.87**	0.80	**0.85**	0.79	0.52
2	0.84	0.27	0.82	0.22	0.53	**0.90**	0.47	**0.88**	0.41	0.51
3	0.87	0.24	0.85	0.22	0.54	**0.91**	0.45	**0.89**	0.44	**0.56**
4	0.82	0.23	0.79	0.18	**0.54**	**0.89**	0.63	**0.87**	0.57	**0.54**

To further quantify memorization and privacy leakage, we report the Attack Success Rate (ASR) of a member inference attack in **Table 1**. Specifically, our approach achieves an ASR of 0.54, placing it close to random guessing. Although Gradient Ascent and Fine-Tuning obtain seemingly favorable ASRs of 0.56 and 0.53, respectively, these results are misleading, as both methods retain substantial predictive power on the forgotten class. Their lower ASRs therefore reflect lingering confidence and membership ambiguity rather than true successful forgetting.

4.3 Representation-Level Evaluation

We next analyze the structural integrity of the learned representations through alignment and uniformity on the retained set (**Table 2**) and inter-class effective rank (**Table 4**). On the retain set, the retrained and fine-tuned models exhibit alignment, uniformity, and inter-class effective rank values that remain close to those of the original pretrained model. This is expected, as both approaches continue to optimize the contrastive loss over retained samples, thereby preserving high inter-class separation. Our method similarly maintains this representational geometry effectively, showing stable uniformity and only a modest anticipated decrease in inter-class effective rank (from 7.50 to 7.17) as the embedding space adjusts to forget the target class. Gradient Ascent, however, substantially disrupts the retain set's structure, heavily increasing alignment and degrading uniformity, indicating poor distribution and excessive geometric distortion.

method demonstrates strong forgetting capabilities, lowering the average accuracy of the forgotten class to 0.19 on the train set and 0.18 on the test set. While Fine-Tuning yields strong retention scores, it critically fails to erase the influence of the forget class, maintaining high accuracies. Conversely, Gradient Ascent demonstrates the lowest utility retention, which likely stems from adversarial nature of the updates, which can interfere with shared features across both retained and forgotten classes, while achieving moderate forgetting.

Table 2. Alignment and Uniformity on the test set of the retained data for the first five classes. The last row reports the absolute average deviation from Retrain. The best and second-best scores are highlighted and underlined respectively.

	Pretrain		Retrain		Our Method		Gradient Ascent		Fine-Tuning	
Class	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**
\0	0.503	-3.657	0.501	-3.680	0.542	-3.648	0.585	-3.370	0.506	-3.677
\1	0.508	-3.658	0.514	-3.686	0.549	-3.653	0.581	-3.390	0.519	-3.679
\2	0.491	-3.665	0.488	-3.679	0.524	-3.649	0.559	-3.504	0.494	-3.677
\3	0.486	-3.662	0.486	-3.680	0.523	-3.651	0.600	-3.510	0.488	-3.676
\4	0.485	-3.660	0.488	-3.680	0.525	-3.650	0.591	-3.503	0.493	-3.676
|Avg. Δ|	0.001	0.021	-	-	<u>0.037</u>	<u>0.031</u>	0.072	0.226	**0.004**	**0.003**

Table 3. Alignment and Uniformity on the test set of the forgotten data for the first five classes. The last row reports the average deviation from Retrain. The best and second-best scores are highlighted and underlined respectively.

	Pretrain		**Retrain**		**Our Method**		**Gradient Ascent**		**Fine-Tuning**	
Class	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**	**Align.**	**Unif.**
0	0.456	-2.684	0.612	-2.749	0.629	-3.345	0.975	-3.608	0.541	-2.628
1	0.384	-2.302	0.641	-2.568	0.730	-3.429	1.092	-3.486	0.522	-2.370
2	0.510	-2.994	0.638	-3.264	0.640	-3.461	0.806	-3.640	0.569	-3.145
3	0.612	-3.058	0.690	-3.082	0.744	-3.483	1.035	-3.656	0.656	-3.038
4	0.543	-2.825	0.691	-2.831	0.733	-3.469	0.996	-3.630	0.602	-2.703
Avg. Δ	0.153	-0.126	-	-	<u>0.041</u>	<u>-0.538</u>	**0.327**	**-0.705**	-0.076	0.122

Table 4. Inter-class effective rank on the test set for the retained data. The last row reports the deviation from Retrain. The best and second-best are highlighted and underlined respectively.

	Pretrain	Retrain	Our Method	Gradient Ascent	Fine-Tuning
Test	7.50±0.06	7.47±0.11	7.17±0.09	6.95±0.19	7.46±0.09
Avg. Δ	0.03±0.08	-	<u>-0.30±0.08</u>	-0.52±0.17	-0.01±0.04

Table 5. Intra-class effective rank on the test set for the forgotten data. The last row reports the deviation from Retrain. The best and second-best are highlighted and underlined respectively.

	Pretrain	Retrain	Our Method	Gradient Ascent	Fine-Tuning
Test	18.50±5.58	24.07±3.51	26.43±2.37	43.55±2.16	22.06±4.17
Avg. Δ	-5.57±2.97	-	<u>2.36±2.68</u>	**19.48±1.80**	-2.01±1.26

Examining the internal coherence of the forget set through alignment and uniformity (**Table 3**) and intra-class effective rank (**Table 5**), we find that retraining from scratch predictably disrupts class-specific structure, reflected by rising alignment and an intra-class effective rank that increases from 18.50 to 24.07. Our method amplifies this effect, exceeding the retrained baseline in representational disruption and raising the intra-class effective rank to 26.43, which indicates a pronounced semantic breakdown within the forgotten class. In contrast, Fine-Tuning leaves much of the original representational structure intact, confirming its limited forgetting efficacy. Gradient Ascent, meanwhile, causes more extreme disruption, nearly doubling the intra-class effective rank and distorting the embedding space entirely beyond what is necessary for balanced unlearning.

1.1 Representation-Level Evaluation

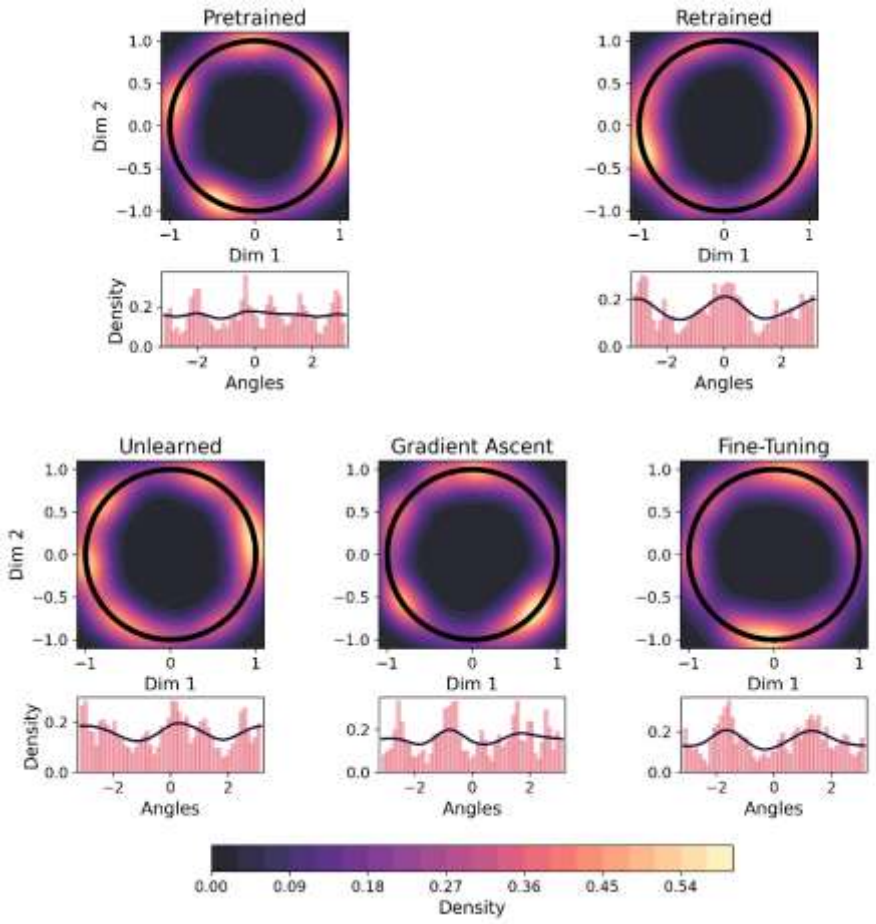


Figure 1: Visualization of the representation space for the forgotten class 5 using 2D t-SNE projections and angular distributions on the unit circle across different unlearning methods.

To qualitatively assess these structural shifts, **Figure 1** visualizes the representation space using 2D t-SNE projections and angular distribution analysis on the unit circle. The pretrained model embeds the forget class with diverse and structured orientations, marked by multiple distinct peaks along the angular spectrum. We observe that the idealized retrained model also shows clustering, but with fewer and broader modes. Both Gradient Ascent and Fine-Tuning exhibit inconsistent alignment with the retrained distributions, often introducing negative symmetries characterized by mirrored peaks on opposing sides of the unit circle. This is expected given their failure to cleanly erase the original structure imparted by the forget set, leading to high variance and

asymmetric spatial concentrations. Our unlearning method, however, consistently produces angular and spatial distributions that closely resemble the geometry of the retrained model. The spatial density contours match the general shape and dispersion patterns of the retrained plots without introducing artificial symmetries, highlighting the robustness of our approach in eliminating the influence of the forget set in a principled manner.

5 CONCLUSION AND FUTURE WORK

In this work, we introduced a principled, proxy-guided framework that addresses the challenging and largely underexplored problem of class-wise forgetting within self-supervised contrastive learning models. Moving beyond superficial downstream task evaluations, our approach operates directly at the representation level. By aligning the embeddings of the forget set with a retain-only proxy and leveraging attention modulation, while simultaneously preserving the structural integrity of the retain set through a contrastive loss, our method achieves an effective balance between forgetting and utility. Our extensive evaluations show that this approach successfully dismantles the semantic coherence of the forgotten class, closely mirroring the representation space of a naively retrained model, while incurring only a modest 5% drop in retain-set accuracy. Furthermore, it effectively scrubs instance-level memorization, significantly reducing the model's vulnerability to membership inference attacks. We further demonstrate that common heuristic baselines, such as Fine-Tuning and Gradient Ascent, either completely fail to induce true forgetting or suffer from severe optimization instability, highlighting the need for a structured, representation-aware unlearning objective.

While our empirical results provide compelling evidence for this approach, they also open exciting new frontiers for the machine learning community. Scaling this framework to large-scale, real-world datasets like ImageNet and diverse architectures such as Vision Transformers (ViTs) represents a critical next step. Additionally, extending this proxy-guided strategy to non-contrastive or clustering-based self-supervised methods, as well as establishing formal

theoretical guarantees for certified unlearning, will further solidify this domain. Ultimately, this research addresses a critical gap in the machine unlearning literature by focusing on the underexplored self-supervised contrastive learning paradigm. Through the development of a novel unlearning framework and a rigorous evaluation protocol centered on representation-level analysis, this work establishes a foundation for future studies in this space and contributes to the broader goal of building more responsible and privacy-aware AI systems.

REFERENCES

[2] Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PmLR.

[24] Caron, M., Bojanowski, P., Joulin, A., & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In Proceedings of the European conference on computer vision (ECCV) (pp. 132-149).

[25] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9729-9738).

[26] Wang, T., & Isola, P. (2020, November). Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In International conference on machine learning (pp. 9929-9939). PMLR.

[27] Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., & Vasilakos, A. V. (2020). Privacy and security issues in deep learning: A survey. IEEe Access, 9, 4566-4593.

[28] Meehan, C., Bordes, F., Vincent, P., Chaudhuri, K., & Guo, C. (2023). Do ssl models have déjà vu? a case of unintended memorization in self-supervised learning. Advances in Neural Information Processing Systems, 36, 42775-42798.

[29] Zaeem, R. N., & Barber, K. S. (2020). The effect of the GDPR on privacy policies: Recent progress and future promise. ACM Transactions on Management Information Systems (TMIS), 12(1), 1-20.

[30] Xu, J., Wu, Z., Wang, C., & Jia, X. (2024). Machine unlearning: Solutions and challenges. IEEE Transactions on Emerging Topics in Computational Intelligence, 8(3), 2150-2168.

[31] Nguyen, T. T., Huynh, T. T., Ren, Z., Nguyen, P. L., Liew, A. W. C., Yin, H., & Nguyen, Q. V. H. (2025). A survey of machine unlearning. ACM Transactions on Intelligent Systems and Technology, 16(5), 1-46.

[32] Shaik, T., Tao, X., Xie, H., Li, L., Zhu, X., & Li, Q. (2024). Exploring the landscape of machine unlearning: A comprehensive survey and taxonomy. IEEE Transactions on Neural Networks and Learning Systems, 36(7), 11676-11696.

[33] Shi, Y., Liu, S., & Wang, R. (2025). Redefining machine unlearning: A conformal prediction-motivated approach. arXiv preprint arXiv:2501.19403.

[34] Tarun, A. K., Chundawat, V. S., Mandal, M., & Kankanhalli, M. (2023, July). Deep regression unlearning. In International Conference on Machine Learning (pp. 33921-33939). PMLR.

[35] Zagoruyko, S., & Komodakis, N. (2016). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint arXiv:1612.03928.

[36] Zbontar, J., Jing, L., Misra, I., LeCun, Y., & Deny, S. (2021, July). Barlow twins: Self-supervised learning via redundancy reduction. In International conference on machine learning (pp. 12310-12320). PMLR.

[37] Bardes, A., Ponce, J., & LeCun, Y. (2021). Vicreg: Variance-invariance-covariance regularization for self-supervised learning. arXiv preprint arXiv:2105.04906.

[38] Oord, A. V. D., Li, Y., & Vinyals, O. (2018). Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

[39] Koromilas, P., Bouritsas, G., Giannakopoulos, T., Nicolaou, M., & Panagakis, Y. (2024). Bridging mini-batch and asymptotic analysis in contrastive learning: From infonce to kernel-based losses. arXiv preprint arXiv:2405.18045.

[40] Cao, Y., & Yang, J. (2015, May). Towards making systems forget with machine unlearning. In 2015 IEEE symposium on security and privacy (pp. 463-480). IEEE.

[41] Ginart, A., Guan, M., Valiant, G., & Zou, J. Y. (2019). Making ai forget you: Data deletion in machine learning. Advances in neural information processing systems, 32.

[42] Graves, L., Nagisetty, V., & Ganesh, V. (2021, May). Amnesiac machine learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 13, pp. 11516-11524).

[43] Fan, C., Liu, J., Zhang, Y., Wong, E., Wei, D., & Liu, S. (2023). Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. arXiv preprint arXiv:2310.12508.

[44] Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., et al. (2021, May). Machine unlearning. In 2021 IEEE symposium on security and privacy (SP) (pp. 141-159). IEEE.

[45] Chen, M., Gao, W., Liu, G., Peng, K., & Wang, C. (2023). Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7766-7775).

[46] Kodge, S., Saha, G., & Roy, K. (2024). Deep unlearning: Fast and efficient gradient-free class forgetting. Transactions on Machine Learning Research.

[47] Guo, C., Goldstein, T., Hannun, A., & Van Der Maaten, L. (2019). Certified data removal from machine learning models. arXiv preprint arXiv:1911.03030.

[48] Sekhari, A., Acharya, J., Kamath, G., & Suresh, A. T. (2021). Remember what you want to forget: Algorithms for machine unlearning. Advances in Neural Information Processing Systems, 34, 18075-18086.

[49] Zhang, B., Dong, Y., Wang, T., & Li, J. (2024). Towards certified unlearning for deep neural networks. arXiv preprint arXiv:2408.00920.

Facial Landmark Detection

Dimitrios A. Chondrogiannis

ABSTRACT

Facial landmark detection, identifies key facial points to enable applications such as expression transfer, emotion recognition, and face morphing. This survey reviews classic and modern facial landmark detection methods, evaluates five backends: Dlib, OpenFace, FAN, and Mediapipe, alongside an implementation of the transformer-based FaceXFormer, and addresses two research questions: (RQ1) How do classic and modern methods compare in accuracy, speed, and robustness? (RQ2) Which methods are best suited for real-time expression transfer and other applications? The contributions of this work include a comprehensive review, an experimental evaluation of the backends on the 300W dataset, and recommendations for application-specific use cases. The conclusions also underscore the need for diverse and representative databases to ensure robust performance across different demographics, poses, and real-world conditions in proof-of-concept implementations and beyond.

ADVISOR

Theoharis Theoharis, Professor, NKUA

1 INTRODUCTION – DEFINITION AND IMPORTANCE

Facial landmark detection is a cornerstone of computer vision, involving the identification of specific 2D points on a human face, such as the corners of the eyes, the tip of the nose, and the contours of the mouth.

The importance of facial landmark detection stems from its ability to provide precise spatial information about facial structures, which is essential for advanced human-computer interaction and visual processing. In computer vision, accurate landmark detection enhances the performance of emotion recognition systems by capturing subtle expression changes, such as a raised eyebrow or pursed lips. In Augmented Reality, landmark detection enables the overlay of digital content onto faces, creating engaging experiences in applications like Snapchat and Instagram filters, where effects dynamically respond to user movements [2]. By mapping landmarks, these platforms achieve realistic and interactive user experiences, driving their widespread adoption.

However, detecting facial landmarks is technically challenging due to variations in pose, lighting, occlusions, and dynamic expressions [3]. These challenges necessitate robust algorithms capable of generalizing across diverse conditions. This paper surveys classic and modern facial landmark detection methods, evaluating established backends. The methods vary in landmark count (68 to 468), computational efficiency, and suitability for applications like expression transfer.

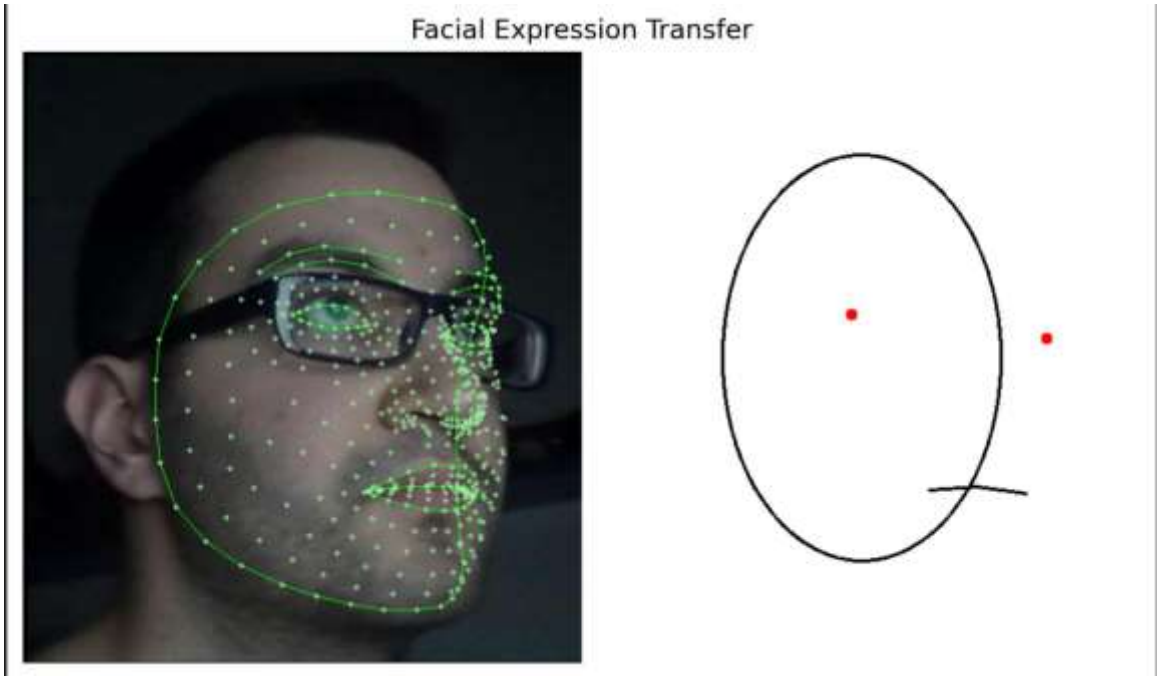
2 APPLICATIONS, APPROACHES, COMPARISONS, LIMITATIONS AND DISCUSSION

2.1 Applications of Facial Landmark Detection

2.1.1 Facial Expression Transfer

Facial expression transfer involves mapping the emotional expressions of a source face onto a target face, enabling realistic animations and virtual interactions. By leveraging landmarks, algorithms capture expression dynamics and apply them to animated characters or digital avatars [4]. In animation and gaming, this technology enhances character realism, allowing figures to mirror human emotions in response to narrative events. In social media, platforms use

landmark detection to create real-time filters that overlay expressions onto avatars, fostering creative user engagement.



Εικόνα 1: Example of facial expression transfer, mapping a source face's expression onto an animated character

2.1.2 Emotion Recognition

Emotion recognition analyzes facial landmark movements to interpret emotions like happiness, sadness, or anger, with applications in psychology, marketing, and user experience design. By tracking changes in landmarks, systems infer emotional states[7]. In marketing, companies use tools to gauge consumer reactions during product testing, refining campaigns based on emotional insights [8]. In mental health, therapists employ landmark-based systems to monitor patients' expressions during sessions, enhancing diagnostic accuracy.

2.1.3 Face Morphing

Face morphing blends facial landmarks to create smooth transitions between faces, widely used in film, visual effects, and social media. By aligning landmarks, algorithms interpolate features to produce effects like character transformations

or aging [2]. For example, films use morphing to depict time progression, while social media platforms offer filters that blend user faces for creative content. However, morphing raises ethical concerns, as deepfake technologies can misuse blended faces for misinformation [10]. Responsible development is essential to balance innovation and societal impact.

2.1.4 Pose Estimation

Pose estimation uses facial landmarks to determine the head's orientation in 3D space, enabling applications in human-computer interaction, gaming, and driver monitoring. By tracking landmarks like the nose tip and eye corners, algorithms compute head angles relative to a camera, facilitating responsive interfaces [11]. In gaming, pose estimation enhances immersion by aligning virtual camera perspectives with player head movements. In automotive safety, systems use landmark-based pose tracking to detect driver attention, reducing accidents caused by distraction [12].

2.2 Key Approaches

Facial landmark detection has evolved from geometric models to data-driven deep learning, driven by the need for robust, accurate localization under diverse conditions. This section surveys key approaches, from early statistical methods to modern transformer-based models, contextualizing the evaluated backends. The survey also traces the field's progression, highlighting technical advancements and their impact on real-world challenges like occlusions, pose variations, and real-time performance.

2.2.1 Early statistical Methods

Early facial landmark detection relied on statistical models to capture facial geometry, laying the foundation for modern techniques. These methods, though limited by rigid assumptions, offer historical context for the field's evolution

These **include Active Shape Models (ASM)** and **Constrained Local Models (CLM).** They were the start of facial landmark detection but struggled with occlusions and pose variations.

2.2.2 Machine Learning Approaches

Machine learning introduced adaptive models to address early methods' limitations, leveraging feature extraction and regression for improved robustness.

These include **Regression-Based Methods and Convolutional Neural Networks (CNNs).** They are fast and accurate, but struggle with extreme variations, and require large datasets for training.

2.2.3 Deep Learning and Heatmap-Based Methods

Deep learning, particularly heatmap-based methods, has significantly improved landmark detection's robustness and accuracy.

These include **Heatmap Regression** and **Hourglass Networks**. They are robust and accurate, but require a lot of computational resources, making them hard to be used in real time.

2.2.4 Hybrid Approaches and Attention Mechanisms

Hybrid and attention-based methods combine multiple techniques to enhance performance in challenging conditions.

These include **Cascade CNNs** and **Attention Mechanisms and Transformers**. They also are robust and accurate, but not suitable for real-time implementations due to costs.

2.2.5 Recent Developments and Future Directions

Emerging techniques address limitations in 2D detection, promising greater scalability and robustness.

These include **3D Landmark Detection** and **Self-Supervised Learning**. Mostly still experimental, but with a promising future.

2.3 Comparison of Methods

This section compares facial landmark detection methods. Building on the survey in Section 2, the performance of these methods is analyzed in terms of accuracy (Normalized Mean Error, NME), robustness to occlusions and pose variations, computational cost, data requirements, and real-time capability, with particular emphasis on suitability for expression transfer. The 300W dataset, comprising 600 indoor and outdoor images annotated with 68 landmarks, provides a standard benchmark for diverse conditions. Experimental results, including NME and inference time, inform recommendations for applications such as animation, gaming, and social media filters.

The strengths, weaknesses, and practical implications of each method are examined. Key takeaways are provided, along with a comparison table based on the implementations developed for each method.

Πίνακας 1: Comparison of facial landmark detection methods. NME (Normalized Mean Error) represents the average landmark localization error normalized by inter-ocular distance, where NME<5%=Very High, 5-7%=High, 7-10%=Moderate, >10%=Low. Robustness indicates failure rate under occlusion, where High <=5%. Pose variability refers to supported yaw rotation, where High = ±60° yaw. Model size approximates computational cost, with low =<100MB and High=> 1GB. Data requirement refers to the approximate size of the training dataset needed for effective performance with small being <=1000 Images**. Real-time indicates performance above 30 FPS.**

Method	Accuracy	Robustness to Occlusion	Pose Variability	Computational Cost	Data Requirement	Real-time
ASM	~10-15%	Poor	Poor (±30°)	Low	Small	Yes
AAM	~8-12%	Poor	Poor (±30°)	Medium	Medium	No
Dlib	~6.5%	Moderate	Moderate (±45°)	Low	Large	Yes
OpenFace	~5.5%	Moderate	High (±60°)	High	Large	No
FAN	~4.0%	High	High (±60°)	Very High	Large	No

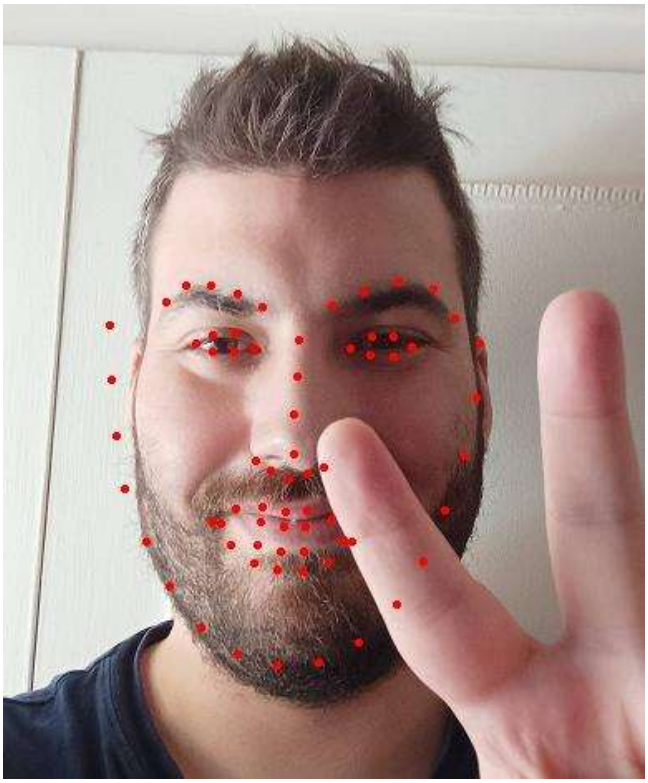
Mediapipe	~3.0%	High	High (±60°)	Medium	Large	Yes
FaceXformer	~3.5%	Very High	Very High (±75°)	High	Large	Yes
3D Detection	~3.5%	Very High	Very High (±75°)	Very High	Large	No

2.4 Challenges and Limitations

Facial landmark detection underpins applications such as expression transfer, animation, and face recognition; however, persistent challenges limit its reliability. This section examines key obstacles, including occlusions, pose variations, expression changes, lighting variations, real-time performance constraints, and dataset limitations, with a focus on their impact on five implemented backends evaluated on the 300W dataset [14]. Drawing on experimental results, the analysis highlights how these challenges affect expression transfer, where precise landmark alignment is critical for realistic animations and social media filters. Solutions grounded in recent research are discussed with the aim of improving robustness and efficiency, while also addressing biases and ethical concerns to enable broader applicability. This section presents the main challenges that facial landmark detection methods must overcome.

2.4.1 Occlusions

Occlusions caused by objects like glasses, hands, or hair, obscure facial features, leading to misaligned landmarks. On 300W, Dlib’s HOG+ERT exhibits a 10% failure rate under occlusions, impacting expression transfer by producing unnatural mouth or eye alignments in animations.



Εικόνα 2: Occlusion Errors

2.4.2 Pose Variation

Extreme poses (±60° yaw) challenge 2D models, as landmarks misalign on non-frontal faces. Dlib and OpenFace show increased NME (~8-10%) for profile views on 300W, distorting expression transfer in gaming avatars.

2.4.3 Expression Changes

Non-rigid deformations from expressions (e.g., smiling, frowning) alter landmark positions, complicating detection. OpenFace's CNNs struggle with subtle AUs, affecting AU-based expression transfer in research.

2.4.4 Lighting Variations

Uneven lighting (e.g., shadows, overexposure) obscures landmarks, increasing NME (e.g., Dlib's ~9% under low light on 300W). This affects video-based expression transfer, causing flickering effects in virtual avatars.

2.4.5 Real-Time Performance

High computational costs limit real-time applications (e.g., live expression transfer). FAN's heatmap regression (8 FPS) and FaceXFormer's transformers (12 FPS) are too slow for mobile devices, unlike Mediapipe's 35 FPS.

2.4.6 Dataset Limitations and Biases

Limited labeled data for diverse populations (e.g., non-Western ethnicities, older ages) reduces model generalization. On 300W, backends show higher NME (~7-10%) for underrepresented groups, raising fairness concerns in expression transfer. Dataset biases can perpetuate inequities.

2.5 Discussion and Further Work

Facial landmark detection has evolved significantly from early statistical models to advanced deep learning frameworks, enabling transformative applications such as expression transfer, emotion recognition, and face morphing. This section discusses the findings in the context of the two research questions posed in the Summary (RQ1 and RQ2).

2.5.1 Research questions answered / Recommendations / Future Directions

This subsection addresses the two central research questions posed in this study ("Comparison of Classic and Modern Methods" and "Suitability for Expression Transfer and Other Applications"), providing insights based on the experimental evaluation. Recommendations for method selection based on application requirements are then presented, followed by a discussion of ethical considerations that should be taken into account. Finally, potential future research directions are proposed to help address the remaining challenges.

3 CONCLUSION

Facial landmark detection has undergone a remarkable transformation, evolving from rudimentary statistical models to sophisticated deep learning frameworks,

enabling a wide array of applications from real-time expression transfer to precise emotion recognition and face morphing. The comprehensive evaluation of the five key backends on the 300W dataset illuminates their distinct strengths and trade-offs in accuracy, speed, and robustness. Mediapipe stands out for its real-time performance, making it ideal for interactive applications like AR filters on platforms such as TikTok and Instagram. FAN and FaceXFormer excel in high-accuracy tasks like offline animation and gaming where capturing fine-grained emotional nuances is critical and OpenFace's Action Unit analysis proves invaluable for research-oriented emotion recognition, despite its slower inference at 12 FPS. These findings underscore the importance of selecting backends tailored to specific application needs, balancing computational efficiency with precision to address challenges like occlusions, pose variations, and dynamic expressions.

Despite these advancements, persistent challenges such as occlusions, dataset biases, and computational constraints continue to limit the reliability of facial landmark detection in unconstrained environments. Dataset biases, particularly underrepresentation of diverse populations, result in higher NME for certain demographics, raising ethical concerns about fairness in applications like social media filters. Emerging solutions, such as self-supervised learning and inclusive datasets, show promise in mitigating these biases, while techniques like model quantization could enhance real-time performance. These developments are critical for ensuring equitable and seamless performance across diverse use cases, from virtual communication to psychological analysis, where accurate landmark detection is paramount.

Looking forward, the field of facial landmark detection is poised for further innovation, with 3D landmark detection and self-supervised learning offering pathways to greater robustness and scalability. Integrating lightweight architectures with occlusion-aware methods could enable real-time applications to handle complex scenarios without sacrificing speed. Additionally, optimizing models for mobile deployment (e.g., achieving 30 FPS through quantization) could bridge the gap between high accuracy and real-time demands. Ethical frameworks and bias-mitigated datasets will be essential to ensure inclusivity, particularly as facial landmark detection shapes human-computer interactions in

animation, gaming, and virtual reality. By addressing these challenges, the field can deliver more reliable, efficient, and equitable solutions, fostering transformative applications that enhance user experiences while upholding fairness and accessibility across diverse populations.

REFERENCES

[1] V. Kazemi and J. Sullivan, “One millisecond face alignment with an ensemble of regression trees,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Columbus, OH, USA, 2014, pp. 1867–1874, doi: 10.1109/CVPR.2014.241.

[2] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, “Face2Face: Real-time face capture and reenactment of RGB videos,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, 2016, pp. 2387–2395, doi: 10.1109/CVPR.2016.262.

[3] A. Bulat and G. Tzimiropoulos, “How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3D facial landmarks),” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Venice, Italy, 2017, pp. 1021–1030, doi: 10.1109/ICCV.2017.116.

[4] C. Cao, Q. Hou, and K. Zhou, “Displaced Dynamic Expression Regression for Real-time Facial Tracking and Animation,” in Proc. ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), Vancouver, BC, Canada, 2014, pp. 43:1–43:10, doi: 10.1145/2601097.2601123.

[5] T. Weise, S. Bouaziz, H. Li, and M. Pauly, “Realtime performance-based facial animation,” ACM Trans. Graph., vol. 30, no. 4, pp. 1–10, Jul. 2011, doi: 10.1145/2010324.1964972.

[6] K. Narayan, V. VS, R. Chellappa, and V. M. Patel, “FaceXFormer: A Unified Transformer for Facial Analysis,” arXiv preprint, arXiv:2403.12960v3, Mar. 10, 2025. [Online]. Available: https://arxiv.org/abs/2403.12960

[7] P. Ekman and W. V. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press, Palo Alto, CA, USA, 1978.

[8] S. Li and W. Deng, “Deep facial expression recognition: A survey,” IEEE Trans. Affect. Comput., vol. 13, no. 1, pp. 119–131, Jan.–Mar. 2022, doi: 10.1109/TAFFC.2020.2981446.

[9] T. Baltrušaitis, P. Robinson, and L.-P. Morency, “OpenFace: An open source facial behavior analysis toolkit,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Lake Placid, NY, USA, 2016, pp. 1–10, doi: 10.1109/WACV.2016.7477553.

[10] R. Chesney and D. Citron, “Deep fakes: A looming challenge for privacy, democracy, and national security,” Calif. Law Rev., vol. 107, pp. 1753–1820, 2019.

[11] E. Murphy-Chutorian and M. M. Trivedi, "Head Pose Estimation in Computer Vision: A Survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, pp. 607-626, May 2009, doi: 10.1109/TPAMI.2008.106.

[12] A. Tawari, K.-H. Chen, and M. M. Trivedi, “Where is the driver looking: Analysis of head, eye and iris for robust gaze zone estimation,” in Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), Qingdao, China, Oct. 2014, pp. 988–994.

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Proc. Int. Conf. Mach. Learn. (ICML), Vienna, Austria, 2020, pp. 1597–1607.

[14] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 faces in-the-wild challenge: The first facial landmark localization challenge,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), Sydney, Australia, 2013, pp. 397–403, doi: 10.1109/ICCVW.2013.59.

[15] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, “Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW), Barcelona, Spain, 2011, pp. 2144–2151, doi: 10.1109/ICCVW.2011.6130513.

[16] K. Karkkainen and J. Joo, “FairFace: Face attribute dataset for balanced race, gender, and age,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Waikoloa, HI, USA, 2021, pp. 1548–1558, doi: 10.1109/WACV48630.2021.00159.

[17] A. Mollahosseini, B. Hasani, and M. H. Mahoor, “AffectNet: A database for facial expression, valence, and arousal computing in the wild,” IEEE Trans. Affect. Comput., vol. 10, no. 1, pp. 18–31, Jan.–Mar. 2019, doi: 10.1109/TAFFC.2017.2740923.

[18] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," Proceedings of Machine Learning Research, 81, pp. 1–15, 2018.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv preprint arXiv:1704.04861v1, Apr. 17, 2017.

OverHAuL: Harnessing Automation for C Libraries with Large Language Models

Konstantinos Chousos

ABSTRACT

Software vulnerabilities remain pervasive in legacy C codebases, where memory safety bugs are persistent and manual fuzzing harness development is costly. This paper presents OverHAuL, a neurosymbolic AI system that employs a triplet of ReAct LLM agents to automatically generate fuzzing harnesses directly from C library source code, requiring no pre-existing harnesses, client code, or documentation. An iterative compilation–evaluation feedback loop refines each generated harness in real time using a function–level codebase oracle for semantic code retrieval. Evaluated on a benchmark of ten open-source C libraries, OverHAuL achieves an 81.25% success rate in producing crash-finding harnesses, demonstrating strong potential for automated vulnerability discovery in previously unfuzzed codebases.

Keywords: fuzzing · LLM agents · harness generation · neurosymbolic AI · software security

ADVISOR

Thanassis Avgerinos, Assistant Professor, NKUA

1 INTRODUCTION

Modern society’s reliance on software systems continues to grow in mission-critical domains—healthcare, aerospace, and industrial infrastructure—where failures can be catastrophic. A significant portion of this foundational software is

written in C, whose lack of memory safety safeguards is notorious. Memory safety bugs remain a persistent vulnerability, and producing provably safe C code is exceptionally challenging. While memory-safe alternatives like Rust and Ada exist, legacy C codebases are not disappearing anytime soon.

Large Language Models (LLMs) have become widely used for code generation, yet frequently introduce subtle errors that evade human review, arguably increasing the prevalence of insecure code [1]. This makes automated vulnerability detection more urgent than ever. *Fuzzing*—executing a program with large numbers of semi-random inputs to expose undefined behavior—is a proven technique for this purpose [2], responsible for high-profile discoveries such as Heartbleed (CVE-2014-0160) and Shellshock (CVE-2014-6271). However, fuzzing requires manually written *harnesses* that exercise a library's API, and authoring effective ones demands deep familiarity with the target codebase—a substantial barrier, particularly for legacy software. Automated harness generation via LLMs has gained traction, but existing approaches uniformly require auxiliary resources beyond the target source code, limiting their applicability to well-maintained projects [3–7].

This paper introduces **OverHAuL** (**H**arness **Au**tomation with **L**LMs), a system that accepts only the source code of a previously unfuzzed C library and autonomously generates an effective fuzzing harness through a triplet of ReAct LLM agents [8] operating in an iterative feedback loop. OverHAuL delegates target function selection to the LLM, requires no external artifacts, and is distributed as a self-contained Python package. Evaluated on ten open-source C libraries, it achieves an **81.25% success rate** in producing crash-finding harnesses.

1.1 Contributions

1. **OverHAuL**: a fully automated end-to-end fuzzing harness generation framework using a triplet of ReAct LLM agents, an iterative feedback loop, and a codebase oracle for semantic source exploration.

2. **Empirical validation** on ten real-world open-source C projects, achieving an **81.25% success rate** in producing defect-uncovering harnesses, with no uncompilable harnesses observed.
3. **Full open-sourcing** of all artifacts and code at https://github.com/kchousos/OverHAuL.

2 BACKGROUND AND RELATED WORK

2.1 Fuzz Testing

Fuzzing is an automated software testing technique in which a Program Under Test (PUT) is executed with pseudo-random inputs in the hope of exposing undefined behavior. When such behavior manifests as a crash, hang, or memory-safety violation, the corresponding input constitutes a test case that reveals a bug and often a vulnerability [2]. Fuzzing is the execution of a PUT using inputs sampled from an input space that protrudes the expected input space of the PUT. When conducted systematically with the goal of detecting security policy violations, it is known as *fuzz testing*. Central to this process is the *fuzzer engine*, which orchestrates execution, and the *bug oracle*—a component that determines whether a given execution violates a security policy, typically realized through sanitizers like AddressSanitizer.

Fuzzing's practical impact is well-established. *Heartbleed* (CVE-2014-0160), a buffer over-read in OpenSSL undetected for two years, would have been revealed almost immediately by a simple fuzz campaign [9]. *Shellshock* variants in GNU Bash were discovered using the AFL fuzzer [10]. The Mayhem security tool [11], adopted by the US Air Force and Cloudflare, has remediated thousands of vulnerabilities across major open-source projects [12].

OverHAuL uses **LibFuzzer** [13], an in-process, coverage-guided engine from the LLVM ecosystem, preferred for its suitability for library fuzzing. LibFuzzer operates through a *fuzz target*—also called a fuzzing harness or fuzz driver—a function conforming to:

```
int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
  DoSomethingInterestingWithData(Data, Size);
  return 0;
}
```

Authoring an effective harness requires deep familiarity with the target library's API, internal structures, and expected usage patterns. This upfront cost is the primary barrier to fuzzing adoption, and the problem OverHAuL directly addresses.

2.2 Large Language Models

LLMs represent the latest milestone in NLP, built on the attention mechanism and Transformer architecture [14]. Key models include the GPT series, Claude, Gemini, DeepSeek-R1, and the Llama series. These fall into closed-source and open-weights categories, the latter offering greater transparency.

Interaction with LLMs occurs through *prompting*, ranging from zero-shot to few-shot approaches. Advanced techniques include Chain of Thought (COT) [15], Tree of Thoughts [16], and Retrieval-Augmented Generation (RAG) [17], which enhances accuracy by querying external knowledge stores in real time. The **ReAct** framework [8] is particularly relevant to this work: it empowers LLMs to act as agents that interleave reasoning with actions on external tools, enabling dynamic interaction with their environment and reducing hallucinated responses.

LLMs have significantly impacted software development through tools like GitHub Copilot and Cursor, giving rise to *vibecoding*—development through prompting alone [18]. However, LLMs remain prone to *hallucination* [19] and are constrained by limited context windows [20], requiring careful validation of generated code. Applied to fuzzing specifically, zero-shot harness generation is ineffective [21]: LLMs over-rely on training patterns, frequently misuse APIs, and introduce errors that shift the verification burden onto developers.

2.3 Neurosymbolic AI

Neurosymbolic AI combines the pattern recognition of neural networks with the structured reasoning of symbolic systems [22], addressing LLM limitations in explainability, attribution, and reliability [23]. OverHAuL instantiates this paradigm as a Neuro[Symbolic] tool per Kautz's taxonomy [24]: the LLM agents leverage a symbolic source code exploration tool—the codebase oracle—to augment their reasoning, rather than relying solely on pattern-matched generation.

2.4 Related Work

Automated harness generation has been approached from several directions, categorized here by the external resources required.

Requiring client code or unit tests. FUDGE [4] is a closed-source Google tool that generates harnesses by slicing existing client code from Google's internal repositories using Clang's AST. FuzzGen [5] constructs an Abstract API Dependence Graph (A^2DG) from client code to synthesize LibFuzzer-compatible harnesses. UTopia [6] operates on user-provided unit tests, treating them as correct API usage examples. All three operate in a linear, feedback-free pipeline and require human specification of the target function—FUDGE's paper explicitly states that "choosing a suitable fuzz target (still) requires a human".

Requiring pre-existing harnesses. OSS-Fuzz [25] is Google's continuous cloud fuzzing platform, having uncovered over 10,000 vulnerabilities across 1,000+ projects since 2016. Acceptance requires a significant user base and considerable infrastructure setup. OSS-Fuzz-Gen (OFG) [3] builds on this platform, using LLMs to generate improved harnesses for already-integrated projects, with coverage gains in projects such as tinyxml2 going from 38% to 69% line coverage [26]. However, OFG is tightly coupled to OSS-Fuzz, difficult to run standalone, and operates without feedback. An experimental from-scratch mode for unfuzzed projects was introduced in 2024 [27] but appears abandoned; in practice, OFG frequently produces malformed outputs—harnesses wrapped in Markdown, missing headers, or using undeclared functions.

Requiring documentation. AutoGen [7] generates harnesses from source code combined with library documentation, with a compilation feedback loop between two LLM instances. The user must specify the target function, and prompt templates—rather than agent exploration—drive generation.

Analysis-based approaches. IntelliGen [28] uses static LLVM analysis to identify vulnerable entry points and synthesize harnesses. CKGFuzzer [29] constructs a code knowledge graph via AST parsing and inter-procedural analysis, using LLMs to generate and repair harnesses for identified API combinations. Both rely primarily on programmatic analysis rather than LLM reasoning, limiting their adaptability to novel codebases.

OverHAuL requires only library source code. It autonomously selects the target function, introduces an iterative multi-agent feedback loop, and uses a function-level vector store oracle for semantic code retrieval—none of which are present in any prior system.

3 THE OVERHAUL SYSTEM

OverHAuL (**H**arness **Au**tomation with **L**LMs) is a neurosymbolic AI tool that automatically generates fuzzing harnesses for C libraries using a triplet of ReAct LLM agents [8] and a codebase oracle containing the project's analyzed source code. Given only a git repository URL, OverHAuL produces a fuzzing harness, a compilation script, a representative crash input, and an execution log—requiring no pre-existing harnesses, client code, documentation, or manual function specification. It is distributed as a self-contained Python package with Clang as its only external dependency, contrasting with comparable systems that require extensive platform integration [3, 25].

3.1 Architecture

OverHAuL's pipeline is divided into three stages: project analysis, harness creation, and harness evaluation. These are orchestrated by a shared iteration

budget of $N = 10$, decremented each time a corrective step is taken. Figure 1 illustrates the full workflow.

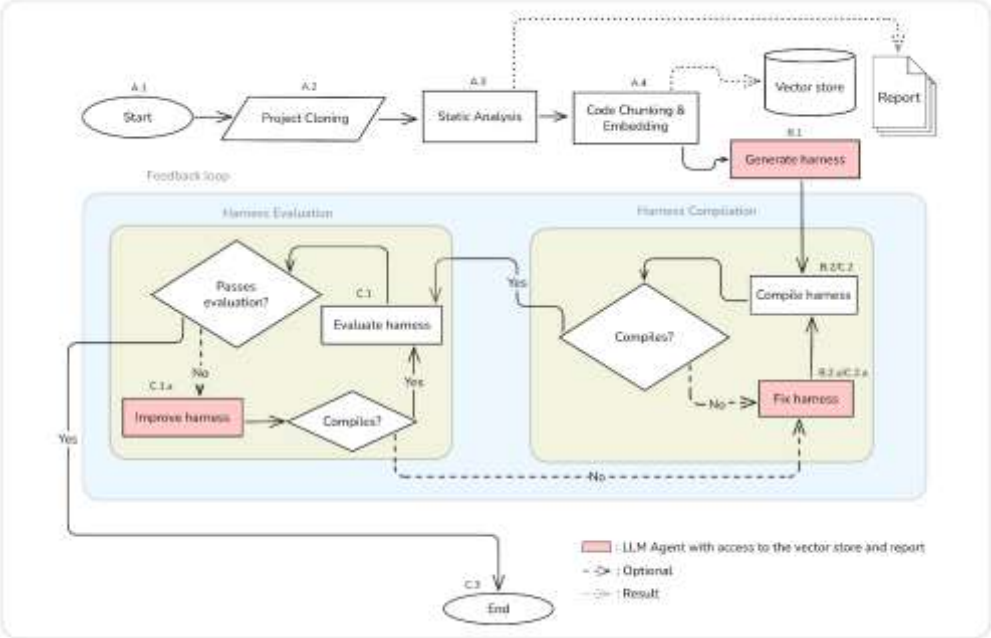


Fig. 1. OverHAuL's iterative workflow. Dashed arrows represent feedback steps.

3.1.1 Project Analysis

The target project is first run through Flawfinder, a static analysis tool that produces a report highlighting potentially vulnerable functions—unsafe uses of strcpy, unchecked memcpy calls, and similar patterns. This report provides the generator agent with meaningful starting points for exploration.

In parallel, the project's source code is parsed using Clang's Abstract Syntax Tree (AST) to extract function-level chunks. Each chunk comprises the function's signature, body, and file path. These chunks are embedded using OpenAI's text-embedding-3-small model and stored in a FAISS vector store index, forming the *codebase oracle*. Files and directories containing "test", "main", "example", "demo", or "benchmark" in their paths are excluded, both to avoid main() conflicts during compilation and to reduce noise in the oracle.

3.1.2 Harness Creation

A “generator” ReAct agent receives the static analysis report and queries the codebase oracle via semantic similarity search, retrieving the top $k = 5$ most relevant function chunks per query. This allows the agent to explore the codebase semantically—for example, querying “functions containing strcpy()”—and identify potentially vulnerable API usage patterns without exhausting its context window.

The harness produced by the generator is compiled using Clang, linked against AddressSanitizer, LeakSanitizer, and UndefinedBehaviorSanitizer. The compilation command is generated programmatically, incorporating all C source files in the project’s root and relevant subdirectories. If compilation fails, the faulty harness and its compiler output are passed to a “fixer” agent, which repairs the errors iteratively until a compilable harness is obtained. The resulting compilation command is exported as a shell script in the project directory.

3.1.3 Harness Evaluation

A compiled harness passes evaluation if and only if: (1) no memory leaks are detected, inferred by the absence of leak-<hash> files; (2) a non-empty crash file is produced, or the harness executes for at least MIN_EXECUTION_TIME without crashing on its own; and (3) the crash file is non-empty, verified via xxd. Condition (2) guards against harnesses that crash immediately due to their own errors rather than library defects—a critical distinction, as developers must be confident that detected crashes stem from vulnerabilities in the tested software rather than flaws inadvertently introduced by the harness.

Harnesses that fail evaluation are forwarded to an “improver” agent, which refines them based on their code and the cause of failure. If an improved harness fails to compile, the fixer agent is re-engaged. All crash files and execution logs are saved in the project directory.

3.2 Key Techniques

3.2.1 Iterative Feedback Loop

The initial harness generated by OverHAuL is unlikely to be correct from the outset. The iterative feedback loop enables systematic refinement under real-world conditions, mirroring the workflow a developer would follow when writing and testing a fuzz target. The shared iteration budget spans both the compilation and evaluation phases, ensuring resource consumption remains bounded while maximizing the chance of producing a valid harness. As demonstrated in the evaluation, this loop is the single most impactful design decision: a one-shot variant of OverHAuL achieves only 28.75% success, compared to 81.25% with full iteration.

3.2.2 ReAct Agent Triplet

Each of the three agents—generator, fixer, and improver—is a separate LLM instance with an independent context, all backed by the same underlying model. Independent contexts serve two purposes. First, they allow broader exploration: the improver agent can investigate strategies the generator may have overlooked or dismissed. Second, they distribute the cognitive load across instances, mitigating the risk of exceeding context window limits on larger codebases. All agents are implemented using the DSPy framework [30], a declarative Python library for LLM programming developed by Stanford's NLP group, chosen over alternatives like LangChain and LlamaIndex for its cleaner abstractions and more reliable behavior [31]. Each agent is a DSPy ReAct module instance with a custom Signature that encodes its role and constraints.

3.2.3 Codebase Oracle

The codebase oracle is the central mechanism by which OverHAuL overcomes the context window limitations that make naive approaches—such as concatenating all source files into a prompt—infeasible for non-trivial projects. By chunking the codebase at the function level and storing embeddings in a FAISS vector store, the oracle enables semantically meaningful retrieval: agents

query it in natural language and receive the most relevant function implementations in response.

The choice of function-level granularity is deliberate. File-level chunking produces chunks that are too large, introducing noise and diluting retrieval quality [32]. Sub-function chunking loses structural context. The function represents the optimal balance between granularity and semantic significance—a principle also recognized in the training of code-specific LLMs, where function-level approaches have been shown to enhance performance. This design is, to the best of our knowledge, novel in the context of automated fuzzing harness generation.

3.3 Implementation

The codebase oracle uses the libclang Python package for AST-based function extraction. Repository cloning uses --depth 1 to minimize disk usage. The current implementation totals 1,254 lines of Python. All benchmark runs are executed via GitHub Actions workflows and are fully reproducible; artifacts are available at https://github.com/kchousos/OverHAuL/actions/workflows/benchmarks.yml. OverHAuL currently targets medium-sized C libraries; C++ support and build system integration (e.g. Make, CMake) are outside the current scope.

4 EVALUATION

To assess the performance and effectiveness of OverHAuL, we structure our evaluation around four research questions:

- **RQ1**: Can OverHAuL generate working harnesses for unfuzzed C projects?
- **RQ2**: What characteristics do these harnesses have? Are they similar to man-made harnesses?
- **RQ3**: How do LLM usage patterns influence the generated harnesses?
- **RQ4**: How do different symbolic techniques affect the generated harnesses?

4.1 Benchmark Setup

To evaluate OverHAuL, a benchmarking script was implemented and a corpus of ten open-source C libraries was assembled. The corpus includes dvhar's dateparse library—also used as a running example in OSS-Fuzz-Gen's experimental from-scratch feature [3]—and nine libraries randomly selected from the package catalog of Clib [33, 34], a package manager for C. All projects are listed in Table 1.

Benchmarks ran from June 6 to July 18, 2025, using OpenAI's gpt-4.1-mini model [35], with a 5-minute harness execution timeout and an iteration budget of 10. Each run was executed as a GitHub Actions workflow on Linux virtual machines with 4 vCPUs and 16 GiB of memory hosted on Microsoft Azure. All reported crashes were manually validated to confirm they stem from genuine library defects rather than harness errors.

Table 1. Benchmark corpus as of July 18, 2025.

Project	Description	Stars	SLOC
dvhar/dateparse	Date parsing without known format	2	2272
clibs/buffer	String manipulation library	204	354
jwerle/libbeaufort	Beaufort cipher implementation	13	321
jwerle/libbacon	Baconian cipher implementation	8	191
jwerle/chfreq.c	Character frequency computation	5	55
jwerle/progress.c	Terminal progress bar library	76	357
willemt/cbuffer	Circular buffer implementation	261	170
willemt/torrent-reader	Torrent-file reader library	6	294
orangeduck/mpc	Parser combinator library	2753	3632
h2non/semver.c	Semantic version parsing library	190	608

4.2 Results

Benchmark outcomes are illustrated in Figure 2, with an iteration heatmap in Figure 3. We now address each research question in turn.

RQ1: Can OverHAuL generate working harnesses for unfuzzed C projects?

OverHAuL achieved an **81.25% success rate** across benchmark runs, producing crash-finding harnesses for previously unfuzzed C libraries. All generated harnesses were valid C programs—a notable contrast with OSS-Fuzz-Gen, which occasionally outputs raw LLM markdown or includes explanatory text inside generated .c files. Crucially, no uncompilable harnesses were observed across any benchmark run, underscoring the effectiveness of the compilation feedback loop and the fixer agent. The harnesses consistently utilized existing functions retrieved from the codebase oracle and interacted appropriately with the target library's API, with only minimal instances of hallucinated or irrelevant code observed.

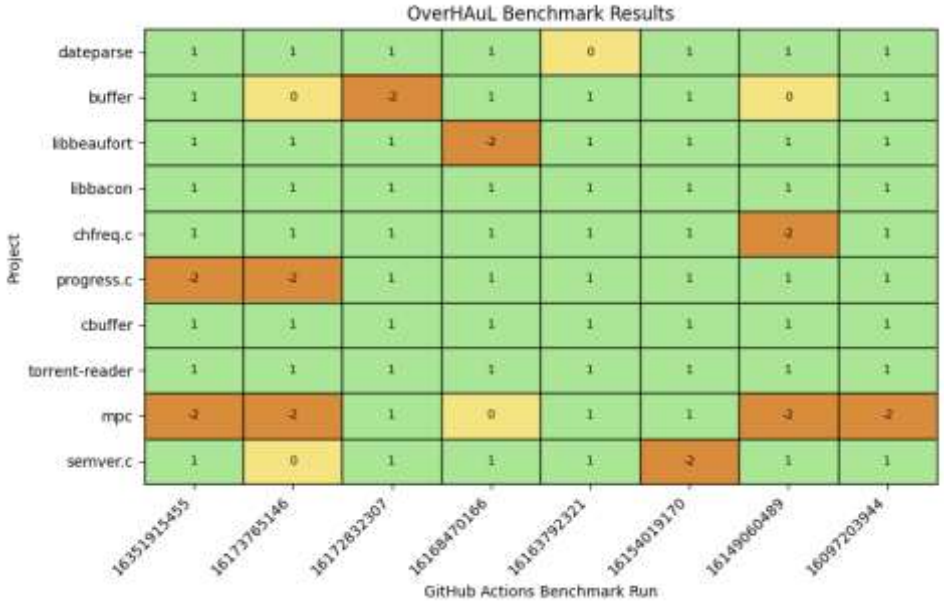


Fig. 2. Benchmark results matrix. Green/1: successful harness with crash input found. Yellow/0: compilable harness, no crash within timeout. Orange/−2: crash attributable to harness error. No red/−1 (uncompilable harness) blocks were observed.

RQ2: What characteristics do these harnesses have? Are they similar to man-made harnesses?

Generated harnesses are typically well-commented, a result of explicit instructions embedded in the generator agent's DSPy Signature. They target both high-level entry-point functions and more narrowly scoped internal functions, depending on what the oracle retrieval surfaces as most vulnerable. In most cases the generated fuzz targets are clear and closely resemble harnesses a skilled software engineer might write, making appropriate and sensible use of the target API. However, some harnesses exhibit unexplained constants or idiosyncratic control flow constructs that can hinder comprehensibility and occasionally introduce errors. The characteristics of generated harnesses also vary substantially across projects and between runs, with differences evident in both size and complexity. Overall, while generated harnesses often echo the structure and intent of man-made harnesses, occasional inexplicable design choices distinguish them from manually written counterparts.

RQ3: How do LLM usage patterns influence the generated harnesses?

Model choice significantly affects results. Both gpt-4.1 and gpt-4.1-mini achieved comparable, strong performance. gpt-4o yielded average results, while gpt-4 and gpt-3.5-turbo averaged only 2 out of 10 successfully harnessed projects per run—effectively making them unsuitable for this task. These findings underscore that gpt-4o represents a minimum baseline for acceptable performance, and that ongoing LLM improvements will directly benefit OverHAuL without requiring system redesign.

ReAct prompting proved decisively superior to the alternatives evaluated. Zero-shot and Chain-of-Thought (COT) [15] approaches were both empirically tested and failed to deliver satisfactory outcomes. The core reason is that effective harness generation requires continuous interaction between the LLM and the target environment—querying the oracle, observing compilation results, incorporating runtime feedback—which COT, as a single-pass technique, cannot support. Agentic workflows are inherently better suited to this class of task.

Local benchmarking further demonstrates a near-linear increase in success rate with iteration budget, confirming the value of the fixer and improver agents. As visible in Figure 3, projects such as dateparse and semver.c exhibit marked improvements when afforded larger iteration budgets, highlighting how iterative refinement recovers from failures that would terminate a one-shot pipeline entirely.

RQ4: How do different symbolic techniques affect the generated harnesses?

Three increasingly capable approaches to code exploration were evaluated during OverHAuL's development. First, direct source code concatenation into prompts failed for all but the smallest projects, as it rapidly exhausted the LLM's context window. Second, a dual-tool approach using an index_tool (returning a JSON project structure) and a read_tool (returning file contents truncated to 4,000 characters) improved scalability but remained constrained: the LLM could only navigate the codebase at the file level, and the character limit left portions of larger files inaccessible. Third, the function-level vector store oracle resolved both limitations, enabling semantically meaningful retrieval at a granularity that neither overwhelms the context window nor loses structural context.

The impact of the iterative feedback loop is equally clear. Earlier one-shot versions of OverHAuL achieved a success rate of only 28.75%. In the current implementation, 42 out of 65 successfully fuzzed projects (64.62%) in the full benchmark did not produce a passing harness on the first attempt and benefited from iteration, with two projects requiring the full eight-iteration budget. This confirms that the feedback loop is not merely a safety net but a core driver of OverHAuL's effectiveness.

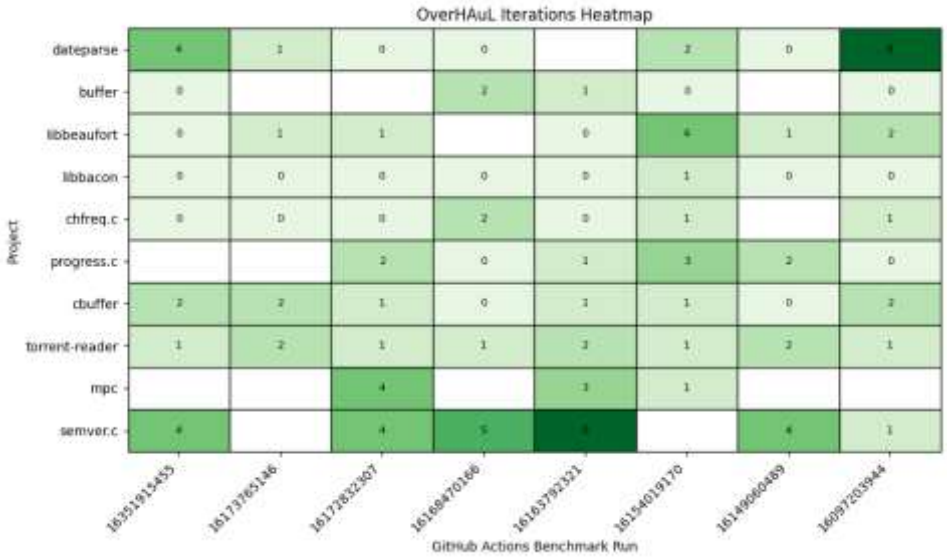


Fig. 3. Iteration heatmap for successfully fuzzed projects. Higher color intensity indicates more iterations needed. Blank cells indicate no valid harness was generated.

4.3 Discussion

OverHAuL's modular architecture means that improvements in LLM capability translate directly into improved harness generation without system redesign. A potential confound in the benchmark is that some evaluated libraries may appear in the LLM's training data, possibly inflating results relative to fully proprietary or unseen codebases. This limitation could theoretically be addressed through targeted fine-tuning. The ten-library corpus also limits generalizability to larger or more architecturally complex projects. Finally, LLM hallucination remains an internal validity threat, though OverHAuL's non-deterministic nature allows reruns to recover from failures, limiting the impact to efficiency rather than correctness.

5 FUTURE WORK

Several directions could meaningfully extend OverHAuL's capabilities.

Build system and language support. OverHAuL currently generates its compilation command programmatically, limiting it to projects that do not rely on build systems. Adding support for Make, CMake, Meson, and Ninja would significantly broaden applicability to larger and more complex codebases. C++ support and multi-language fuzzing engines such as Google's Atheris for Python would further widen OverHAuL's reach.

Additional fuzzing engines. Integrating AFL++ [36] or GraphFuzz [37] would diversify available fuzzing strategies. Multi-engine support would also allow coverage-based evaluation metrics—such as differential coverage tracking between harness versions—to replace or complement the current crash-based success criterion, yielding deeper insight into harness quality.

LLM experimentation. Benchmarking alternative models—including OpenAI's o1 and o3 families, Anthropic's Claude Opus 4, and code-focused models such as CodeGen—would clarify capability and cost trade-offs. Experimenting with finer-grained chunking strategies, such as splitting at the sub-function level, could also improve oracle retrieval quality for densely implemented libraries.

Incorporating existing artifacts. OverHAuL currently ignores client code and unit tests when present. Incorporating these as additional oracle sources or generator context—inspired by FUDGE [4] and FuzzGen [5]—could yield more targeted harnesses for projects where such resources exist, without making them a hard requirement.

Large-scale evaluation and deployment. Scaling benchmarks to the full Clib catalog [33], which encompasses hundreds of C libraries, and conducting direct comparisons against OSS-Fuzz-Gen [3] and AutoGen [7] under controlled conditions would strengthen validity claims. Packaging OverHAuL as a GitHub Actions workflow would lower adoption barriers for developers wishing to integrate continuous fuzzing into their CI pipelines. Finally, automating pull requests to affected library maintainers—reporting discovered crashes with reproducing inputs—would close the loop between automated vulnerability discovery and open-source software improvement.

6 CONCLUSION

This paper presented **OverHAuL**, a neurosymbolic AI system that autonomously generates fuzzing harnesses for C libraries directly from source code, requiring no client code, pre-existing harnesses, documentation, or manual function specification. By combining a triplet of ReAct LLM agents [8] with a function-level codebase oracle and an iterative compilation–evaluation feedback loop, OverHAuL achieves an **81.25% success rate** on ten open-source C libraries, with no uncompilable harnesses observed across any benchmark run.

Compared to prior work, OverHAuL's distinguishing characteristics are: the elimination of all auxiliary resource requirements; autonomous target function selection by the LLM agent; an iterative multi-agent feedback loop that recovers from both compilation and evaluation failures; and a semantically meaningful vector store oracle that enables codebase exploration without exhausting context window limits. These properties make OverHAuL particularly well-suited to legacy and under-tested C codebases, where pre-existing test infrastructure is absent and manual harness authoring is impractical or costly.

The system is fully open-source and available at https://github.com/kchousos/OverHAuL, with all benchmark artifacts and GitHub Actions workflows available for full reproducibility.

REFERENCES

[1] Perry, N., Srivastava, M., Kumar, D., Boneh, D.: Do Users Write More Insecure Code with AI Assistants?, arxiv.org, (2023). doi.org.

[2] Manes, V.J.M., Han, H., Han, C., Cha, S.K., Egele, M., Schwartz, E.J., Woo, M.: The Art, Science, and Engineering of Fuzzing: A Survey, arxiv.org, (2019). doi.org.

[3] Liu, D., Chang, O., metzman, J., Sablotny, M., Maruseac, M.: OSS-fuzz-gen: Automated fuzz target generation, github.com, (2024).

[4] Babić, D., Bucur, S., Chen, Y., Ivančić, F., King, T., Kusano, M., Lemieux, C., Szekeres, L., Wang, W.: FUDGE: fuzz driver generation at scale. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 975–985. ACM, Tallinn Estonia (2019). doi.org.

[5] Ispoglou, K., Austin, D., Mohan, V., Payer, M.: FuzzGen: Automatic fuzzer generation. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 2271–2287 (2020).

[6] Jeong, B., Jang, J., Yi, H., Moon, J., Kim, J., Jeon, I., Kim, T., Shim, W., Hwang, Y.H.: UTopia: Automatic Generation of Fuzz Driver using Unit Tests. In: 2023 IEEE Symposium on Security and Privacy (SP). pp. 2676–2692 (2023). doi.org.

[7] Sun, Y.: Automated Generation and Compilation of Fuzz Driver Based on Large Language Models. In: Proceedings of the 2024 9th International Conference on Cyber Security and Information Engineering. pp. 461–468. Association for Computing Machinery, New York, NY, USA (2024). doi.org.

[8] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing Reasoning and Acting in Language Models, arxiv.org, (2023). doi.org.

[9] Wheeler, D.: How to Prevent the next Heartbleed, dwheeler.com, last accessed 2025/03/25.

[10] Saarinen, J.: Further flaws render Shellshock patch ineffective, itnews.com.au, last accessed 2025/07/15.

[11] Cha, S.K., Avgerinos, T., Rebert, A., Brumley, D.: Unleashing mayhem on binary code. In: 2012 IEEE symposium on security and privacy. pp. 380–394. IEEE (2012).

[12] Simonite, T.: This Bot Hunts Software Bugs for the Pentagon, wired.com, (2020).

[13] LLVM Project: libFuzzer – a library for coverage-guided fuzz testing. — LLVM 21.0.0git documentation, llvm.org, last accessed 2025/04/08.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need, arxiv.org, (2023). doi.org.

[15] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, arxiv.org, (2023). doi.org.

[16] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T.L., Cao, Y., Narasimhan, K.: Tree of Thoughts: Deliberate Problem Solving with Large Language Models, arxiv.org, (2023). doi.org.

[17] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, arxiv.org, (2021). doi.org.

[18] Sarkar, A., Drosos, I.: Vibe coding: programming through conversation with artificial intelligence, arxiv.org, (2025). doi.org.

[19] Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., Liu, T.: A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. ACM Trans. Inf. Syst. 43, 1–55 (2025). doi.org.

[20] Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., McHardy, R.: Challenges and Applications of Large Language Models, arxiv.org, (2023). doi.org.

[21] Jiang, Y., Liang, J., Ma, F., Chen, Y., Zhou, C., Shen, Y., Wu, Z., Fu, J., Wang, M., Li, S., Zhang, Q.: When Fuzzing Meets LLMs: Challenges and Opportunities. In: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering. pp. 492–496 (2024). doi.org.

[22] Sheth, A., Roy, K., Gaur, M.: Neurosymbolic AI -- Why, What, and How, arxiv.org, (2023). doi.org.

[23] Gaur, M., Sheth, A.: Building Trustworthy NeuroSymbolic AI Systems: Consistency, Reliability, Explainability, and Safety, arxiv.org, (2023). doi.org.

[24] Kautz, H.: The Third AI Summer. 34th Annual Meeting of the Association for the Advancement of Artificial Intelligence. , New York, NY, USA (2020).

[25] Arya, A., Chang, O., Metzman, J., Serebryany, K., Liu, D.: OSS-Fuzz, github.com, (2025).

[26] Liu, D., Metzman, J., Chang, O., Team, G.O.S.S.: AI-Powered Fuzzing: Breaking the Bug Hunting Barrier, googleblog.com, last accessed 2025/07/10.

[27] OSS-Fuzz Maintainers: Introducing LLM-based harness synthesis for unfuzzed projects, oss-fuzz.com, last accessed 2025/03/31.

[28] Zhang, M., Liu, J., Ma, F., Zhang, H., Jiang, Y.: IntelliGen: Automatic Driver Synthesis for FuzzTesting, arxiv.org, (2021). doi.org.

[29] Xu, H., Ma, W., Zhou, T., Zhao, Y., Chen, K., Hu, Q., Liu, Y., Wang, H.: CKGFuzzer: LLM-Based Fuzz Driver Generation Enhanced By Code Knowledge Graph, arxiv.org, (2024). doi.org.

[30] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T.T., Moazam, H., Miller, H., Zaharia, M., Potts, C.: DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, arxiv.org, (2023). doi.org.

[31] Both, F.: Why we no longer use LangChain for building our AI agents, octomind.dev, last accessed 2025/07/18.

[32] Zhao, S., Yang, Y., Wang, Z., He, Z., Qiu, L.K., Qiu, L.: Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to Make your LLMs use External Data More Wisely, arxiv.org, (2024). doi.org.

[33] Clibs Project: Clib Packages, github.com, last accessed 2025/07/05.

[34] Clibs Project: clibs/clib, github.com, (2025).

[35] OpenAI Docs: GPT-4.1 mini - Open AI API, openai.com, last accessed 2025/07/18.

[36] Heuse, M., Eißfeldt, H., Fioraldi, A., Maier, D.: AFL++, github.com, (2022).

[37] Green, H., Avgerinos, T.: GraphFuzz: library API fuzzing with lifetime-aware dataflow graphs. In: Proceedings of the 44th International Conference on Software Engineering. pp. 1070–1081. ACM, Pittsburgh Pennsylvania (2022). doi.org.

Διπλωματικές Εργασίες

Design and Development of a Virtual Reality and Web-Based Training System for Early Detection and Intervention in Psychosis

Konstantinos Theofilis

ABSTRACT

This thesis presents the design, development, and evaluation of a digital training system that combines virtual reality (VR) and web-based technologies to enhance awareness and communication skills among relatives of individuals at risk of psychosis. The system consists of three integrated components: a web authoring application for clinicians to create and manage training scenarios, a VR application for relatives or close individuals to experience these immersive scenarios, and a shared database that connects the environments. The authoring web app allows clinicians to build structured dialogue-based conversations that can be stored in the document-oriented database and linked to the VR experience using unique scenario codes. The VR app allows the user to practice, in a secure and supervised environment, supportive communication and psychosis detection with a virtual NPC. A mixed-method evaluation was conducted. The web application was assessed by seven (N=7) mental health professionals using the System Usability Scale (SUS). The VR application was evaluated by thirty participants (N=30) using the Customisable Interactions Questionnaire (CIQ), with high scores across all five factors. Qualitative feedback highlighted both strengths and areas for improvement. Details of the findings of all the analyses are presented in the final chapters, along with recommendations and adjustments for future researchers.

Keywords: Virtual Reality, Human-Computer Interaction, Psychosis Detection, Mental Health Training, Scenario-Based Learning

ADVISOR
Maria Roussou, Associate Professor, NKUA

1 INTRODUCTION

Psychotic spectrum disorders are among the most serious psychiatric diseases, putting a heavy burden on patients, their families, and the broader community. In Greece, around 3,000 adolescents and young adults are estimated to experience their first psychotic episode annually [1]. In the absence of timely psychosocial intervention and pharmacological treatment, a significant number of these individuals can follow a chronic path marked by relapses, recurrent hospital admissions, and decreased functioning. Research verifies the fact that early multi-aspect and community-based treatment correlates with a better prognosis and a better result for patients with psychotic disorders [1]. Despite prior initiatives, a gap exists for functional tools suitable for the education of non-professionals, most importantly, family members and close associates, regarding the detection of the early warning signs of psychosis and responding empathetically. Established formats, such as written reports and psychoeducational groups, tend to prove too abstract or passive. Virtual Reality (VR) has demonstrated considerable promise across a variety of mental health domains [4, 5, 6]. However, it is evident that most research has focused on the patient, rather than educating carers on how to identify and support individuals who are at risk [4]. This thesis proposes and implements a training system that combines immersive virtual reality (VR) with interactive digital storytelling to support early detection and supportive responding for psychosis. The solution fills a known gap by providing easily accessible resources for non-specialists (family members or close friends) to practice identifying early warning signs and reacting in non-stigmatizing, emotionally safe ways. The system comprises two tightly integrated applications, a Web Application for clinicians, used to author training scenarios as structured, branching narratives, and a VR Application for relatives and close individuals, which executes those scenarios as interactive simulations in a home setting [8, 9]. This thesis was formulated in close

collaboration with the Early Intervention Unit of EPAPSY PNOES Athens, who provided insightful advice, expert knowledge, and clinical understanding of the issue. Several educational programmes have targeted mental health training for clinical and non-clinical audiences. The results of the Schiz'Aids program showed that caregivers' understanding of schizophrenia was improved, their burden lessened, and their depression decreased [3]. The Family-to-Family program showed that after the program was finished, participants showed improved family empowerment, better family functioning, increased health knowledge, improved coping skills, and decreased burden of caregivers [2]. Despite these advances, these approaches remain primarily passive or lecture based. Riches et al. [4] explored VR-based training for mental health staff to increase empathy, compassion, and subjective understanding. Bridge et al. [5] evaluated a VR environment for supporting student well-being on clinical placement. Spytska [6] reviewed VR applications for phobia and PTSD treatment, highlighting advantages including safe exposure, individualised treatment, realistic simulations, and improved accessibility. Chan et al. [7] conducted a systematic review of immersive VR for psychosis assessment and intervention, concluding that VR-based interventions show promise but require further validation. To sum up, it is significant to combine virtual reality technology with easily navigable tools that allow non-technical users to create and modify interactive VR experiences [8, 9]. The overall mean SUS score was M = 81.4, placing the system in the "Excellent" usability range [10]. The results of the CIQ revealed generally high levels of user satisfaction across all five evaluated factors, showing an overall positive user experience within the VR training environment [11]. Notably, several participants reported uncertainty or mild anxiety during dialogue choices, indicating genuine emotional engagement with the simulated scenario. It should be noted that the evaluation is focused on usability and user experience rather than therapeutic or clinical efficacy, as the thesis is framed as a design and prototyping exercise rather than a clinical intervention.

2 METHODOLOGY AND DESIGN

The system was created to address two (2) distinct user groups, defined by different obligations and needs. The first group consists of clinicians (psychologists, psychiatrists, and other mental health professionals) who are responsible for each educational scenario. The second group, who will experience training in VR, consists of relatives or close individuals of people who might need to be more aware of early symptoms of psychosis. The selected methodology combines iterative prototyping and user-centered design (UCD), with elements of co-design applied specifically in collaboration with the first user group (clinicians). To achieve this, we engaged in regular meetings with the Association for Regional Development and Mental Health PNOES to gather requirements and determine precisely how the system should look and behave to provide a better user experience. The second user group (relatives) was not involved in the design phase but was engaged during the evaluation phase. The system was developed focusing on user needs, preferences, and behaviors across every phase of the system's lifecycle.

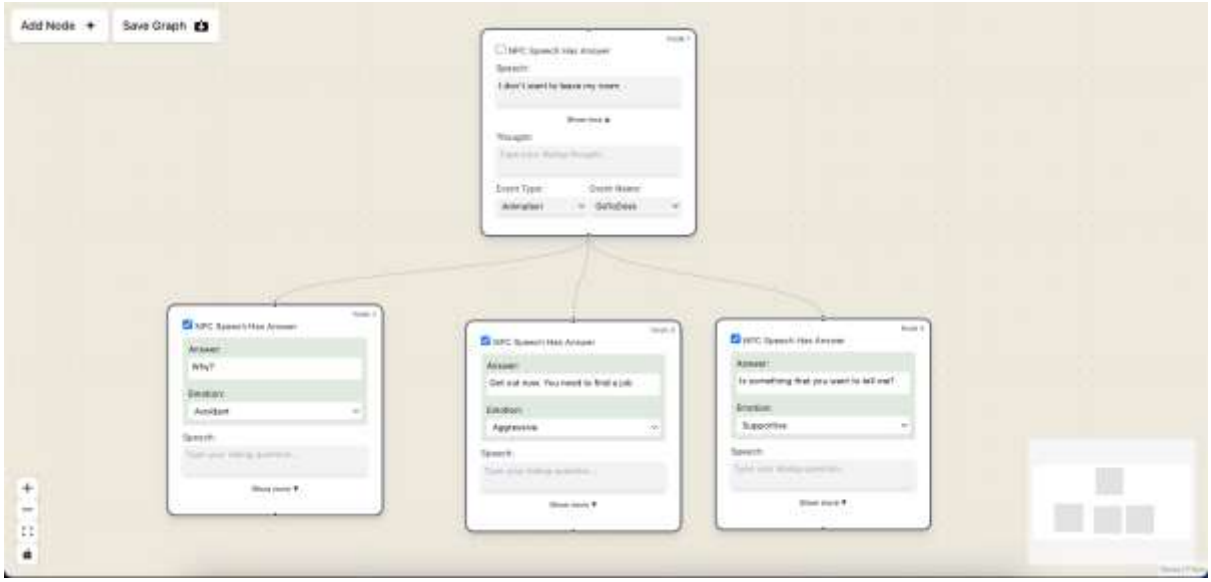
2.1 Defining Training Scenarios

One of the most challenging aspects of this project was defining what exactly a training scenario is, and which are the fundamental building blocks. The definition and structure of training scenarios were developed in close collaboration with the first user group (clinicians). Specifically, four (4) co-design sessions were conducted with three (3) clinicians from EPAPSY PNOES, including psychiatrists and therapists. These sessions focused on identifying realistic early psychosis warning signs, defining appropriate dialogue content, and determining the emotional and behavioral parameters of the NPC. The clinicians contributed directly to shaping the branching narrative structure, the emotional labels for user (relatives) responses, and the selection of animation events. This iterative process ensured clinical validity of the scenario content. Thus, a training scenario was defined as an interactive simulation that combines dialogue, emotion, cognition, and movement to create realistic encounters for learning and practice. Each training scenario is not merely a branching dialogue, but a multi-sided simulation that integrates speech, thought, emotion, and embodied

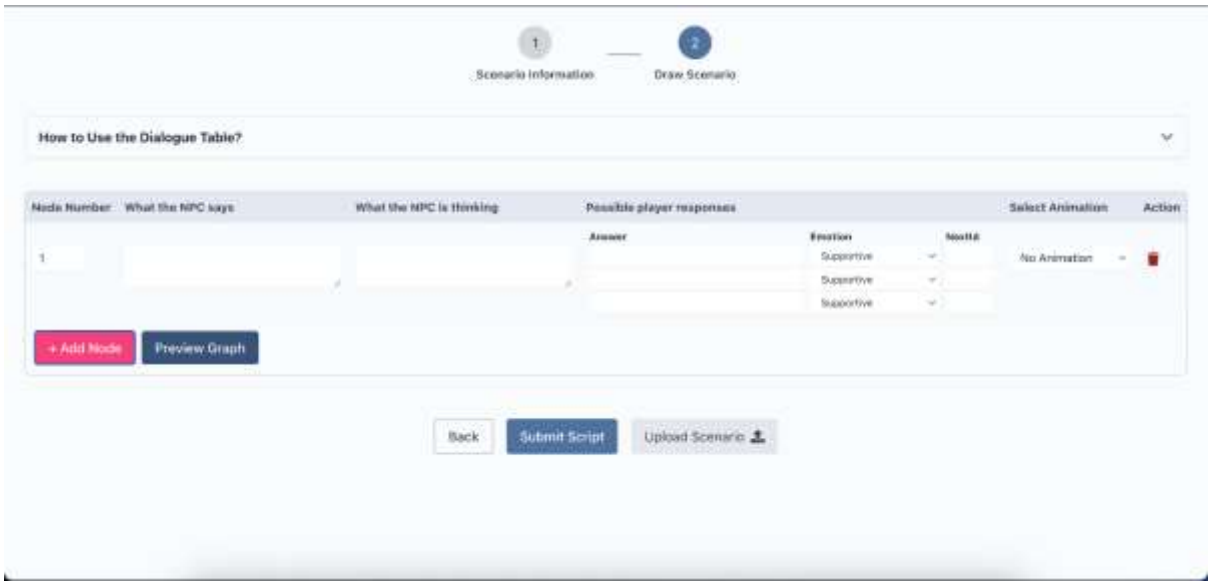
behavior into a cohesive interactive experience. However, a training scenario is composed of smaller, self-contained pieces that create an interaction, and it is called a “dialogue” component. A “dialogue” is a step of interaction between the actor and the digital NPC. The interactive dialogue is modeled as a directed graph $G = (V, E)$ where V is a set of dialogue nodes, and E represents transitions based on player choices. Terminal nodes $\{T = v \in V | v.answers|\}$ mark the conclusion of the interaction. Each dialogue node captures: (a) speech, the POI's spoken line; (b) thought, the POI's internal monologue; (c) event type and name, triggering animations; and (d) three player answers, each associated with an emotion and a link to the next node. This multi-layered design reflects the richness of real human interactions, particularly those involving early psychosis signs.

2.2 Scenario Editor Design

The Scenario Editor design had to balance expressive power with ease of use for clinicians who were not experienced with advanced digital writing tools. The initial version used a graph-based interface where each node represented a step in a dialogue, with fields for speech, thought, emotion, and event definitions. Nodes could connect with drag-and-drop lines, creating a branching structure that visually represented the dialogue flow. Initial testing with clinicians revealed problems, as they were unfamiliar with node editing systems, and the interface became cumbersome as scenarios grew. The large number of editable properties also created unnecessary complexity and cognitive overload.



In response, the editor was redesigned as a table-based interface where each row represents a node and columns represent features such as speech, thought, and emotion. Decisions appear in the same row with identifiers pointing to the next node, preserving the branching conversation flow. The event system was simplified to animation events, reducing complexity while retaining key non-verbal cues. A non-editable graph-based preview mode allows clinicians to visualize the entire scenario and verify the branching structure before submission.



2.3 VR Simulation Design

The VR simulation design was shaped by the structured scenario content and clinical requirements that emerged from the co-design process with clinicians, as well as by established VR usability principles. The second user group (relatives) was not involved in the design of the simulation, and their role was reserved for the evaluation phase. The selected scenario was a family-oriented situation in which the user, represented as a parent, interacts with the digital NPC, portrayed as an adolescent POI who refuses to leave his or her room. The virtual environment was created as a two-story house with a ground and a first floor with the ground floor to act as an introductory space that helps users gradually adapt to the virtual reality environment.

The graphical style of the environment follows a mid-poly aesthetic that prioritizes clarity and accessibility over hyper-realistic detail. Within the VR application, scenarios are selected through a unique code consisting of one capital letter and five numbers, allowing each scenario created through the web application to be uniquely identified. The digital NPC that represents the POI is portrayed by a neutral mannequin-style 3D model designed with a low-poly aesthetic that aligns with the overall visual style of the environment. All interactions and movement in the VR environment are performed using VR controllers, with the control scheme limited to a maximum of three buttons.

Movement through the environment is implemented using anchored teleportation to predefined locations represented by small platforms. Additional usability is provided through visual affordances represented as blue circular buttons with icons indicating available actions. The dialogue system is presented through two main user interface components: the Player's Head UI and the POI's UI. The digital NPC's behavior is controlled through an animation graph that manages transitions between sitting and standing states at three key locations within the bedroom. The circular sequence of animation states, beginning with the digital NPC in a sitting position. To move to another location, the digital NPC first transitions through a stand-up animation, after which the character is placed into the sitting position of the new destination.

3 IMPLEMENTATION

3.1 System Architecture

The system consists of three integrated components. The Web Application (React, Vite, Tailwind CSS, Flowbite), the VR Application (Unity, Unity XR Framework, Firebase, Unity SDK), and the Firestore Database. Clinicians can create, edit, and delete scenarios in the web application. These are then stored in Firestore and retrieved in the VR application via a unique alphanumeric scenario code (format: one capital letter + five numerals, yielding up to 2,600,000 permutations).

3.2 Web Application

The web application is organized into four functional layers. Firstly, the access layer with main responsibility is to manage login and authentication. The scenario authoring layer is equipped with tools to design scenarios that capture dialogues, emotional states, and possible responses from the POI. Preview and visualization layer (read-only graph preview), and finally scenario management layer (CRUD operations on Firestore). Scenarios are stored as JSON documents in Firestore, encoding metadata (title, language, voice, relationship) and a nodes

array representing the dialogue graph. Clinicians can also upload pre-authored JSON files directly into the editor.

3.3 VR Application

The VR application is organized into four layers. Firstly, Access provides the entry point for users to start a training session. The scenario layer (dialogue branching logic, POI behavior), the interaction layer (simplified controller input, teleportation, affordances), and finally the immersion (environment, POI animations, TTS audio). Text-to-Speech synthesis is provided by the ElevenLabs API, with configurable voice styles and reverb differentiation between spoken dialogue (no reverb) and internal thoughts (reverb). The Animation Manager uses a coroutine-based fade-in/fade-out system for POI repositioning, ensuring smooth transitions without full walking animations.

4 EVALUATION AND RESULTS

4.1 Web Application - SUS Evaluation

The web application evaluation included seven (N=7) mental health professionals (psychiatrists and clinical staff) with experience in the illness of psychosis. Their area of expertise served as the selection criterion because the study aimed to assess the web application’s technical features, as well as because the training scenario creation tool is designed for their professional use. Before they began interacting with the web application, participants were given a brief overview of the web application's features and how to use it. To analyse the web application, the System Usability Scale (SUS) questionnaire was used. SUS is a very standardized questionnaire consisting of 10 questions that use the five-point Likert scale in recording the response [10]. The overall mean SUS score was M = 81.4, placing the system in the "Excellent" usability range according to Bangor et al. [10].

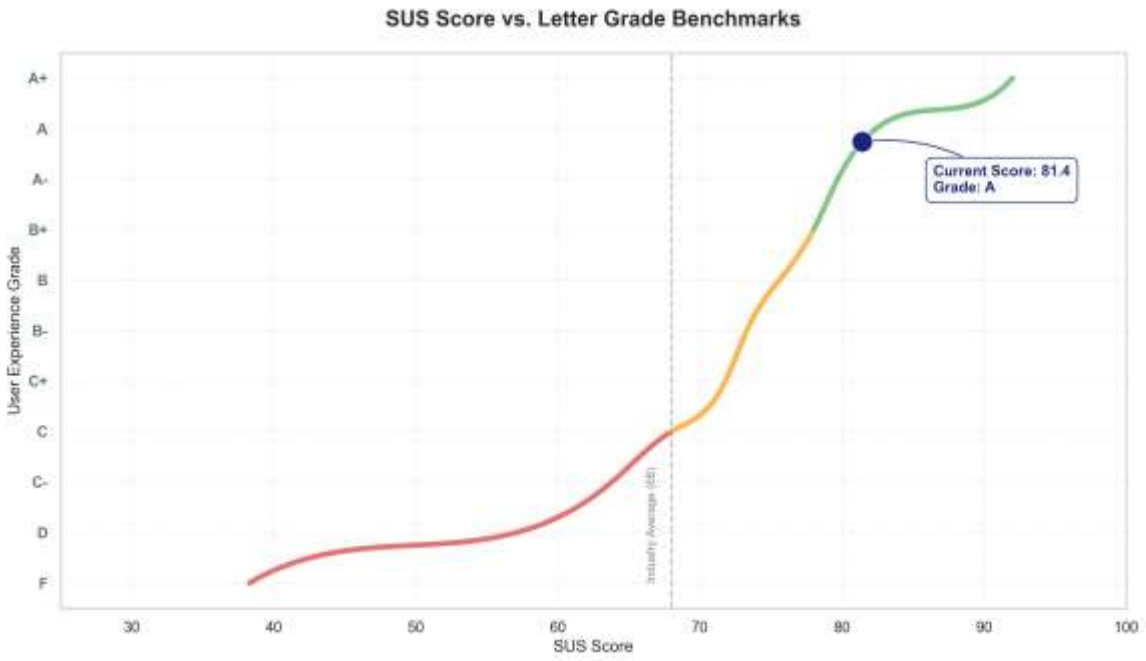


Fig. 1. System Usability Scale (SUS) score of the web application relative to standardized letter grade benchmarks.

Individual scores ranged from 70 to 92.5. All participants exceeded the 68-point satisfactory usability threshold. Item-level analysis revealed that most participants felt confident and engaged with the system. However, most users may want technical support to quickly become proficient with the system.

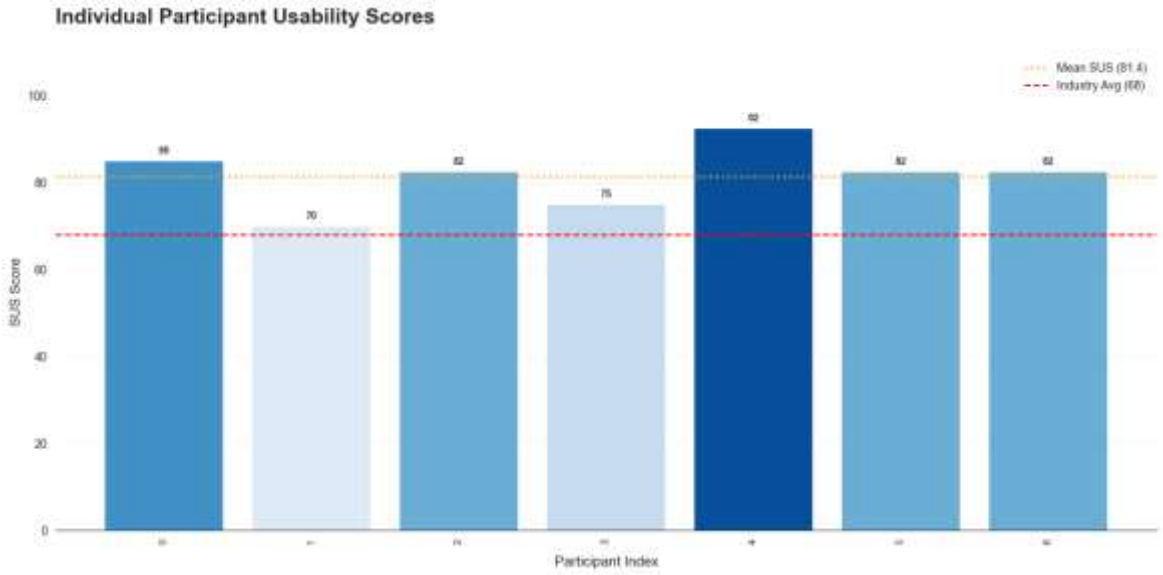


Fig. 2. System Usability Scale (SUS) for each individual participant.

4.2 VR Application - CIQ Evaluation

The VR application evaluation included thirty (N=30) general users. The evaluation involved adult participants of various ages who volunteered to take part in the VR evaluation, and no profession or expertise was required. To perform the quantitative analysis, we employed the Customisable Interactions Questionnaire [11], which is intended to measure users' subjective experiences interacting with virtual or augmented reality environments The results of the CIQ revealed generally high levels of user satisfaction across all five evaluated factors. This shows an overall positive user experience within the VR training environment.

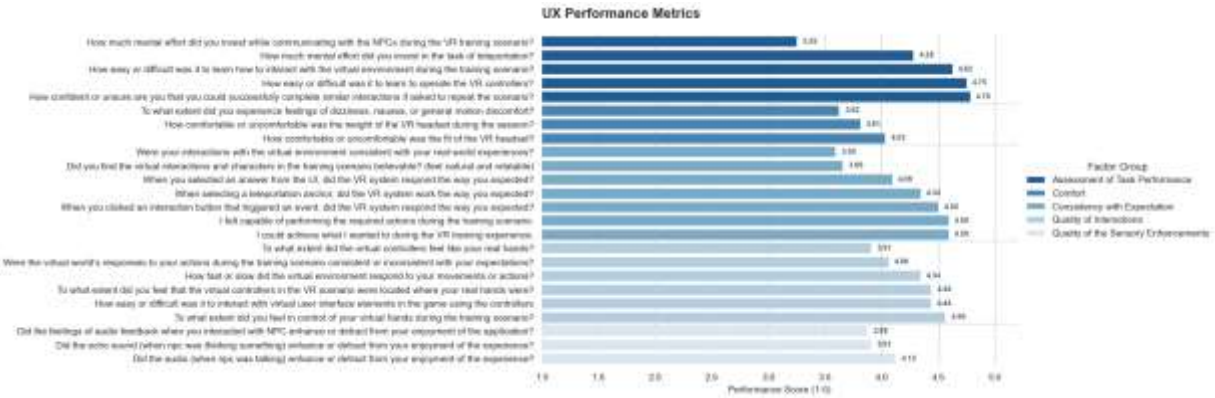


Fig. 3. Average mean score per CIQ question.

Assessment of Task Performance achieved the highest mean score (4.34), suggesting that participants felt confident in their ability to complete all the assigned interactions and effectively navigate the scenario. Consistency with Expectations (4.20) and Quality of Interactions (4.29) followed closely, reflecting that the system's responses and the realism of virtual interactions aligned well with user expectations. Quality of Sensory Enhancements (3.97), although still above average, highlights potential areas for improvement related to visual fidelity, sound feedback, and sensory realism (elements that are often critical for full immersion in VR experiences). Finally, the lowest factor was Comfort (3.82), which also received an above-average score, indicating that most users experienced minimal discomfort while using the VR headset and interacting within the simulated environment.

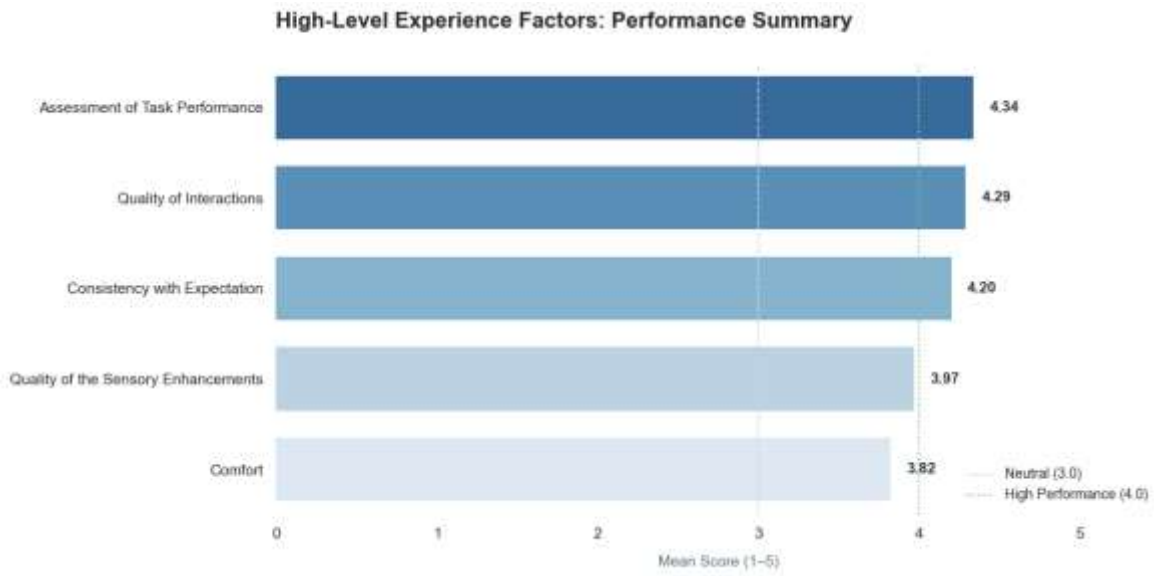


Fig. 4. Average mean score per CIQ factor.

4.3 Qualitative Feedback

A qualitative study was also used as a complement to the quantitative study, although on a voluntary basis. The same participants had the opportunity to express their own opinions regarding the VR application. The participants had to answer an open-ended question, writing a paragraph of positive or negative feedback. The length of the text was limited to 600 characters. In our approach to analyzing these, we used a simple one-level priority scheme because our data is based on observations, comments, and opinions. We categorized these into three buckets: high, medium, and low, depending on how these issues contribute to the experience. The high-priority issues are the ones that will significantly impact usability, engagement, or learning. The medium-priority issues are also important but will not significantly impact the experience. They are not critical, but they should be fixed. The low-priority issues are cosmetic and not critical.

High-priority negative findings included overlapping thought audio and dialogue of UI elements, confusion between NPC thoughts and spoken responses, and AI-generated voice perceived as lacking emotional empathy. Medium-priority issues included NPC slow interaction pacing, desire for longer scenarios and/or dialogue options, dialogue box visual load and trigger placement, and the lack of

pre-experience choices before the main interaction starts. Low-priority issues included the desire for scoring/feedback, a more game-like exploration mode, and slightly faster dialogue speed.

Positive feedback consistently praised the low-poly visual style for supporting focus on content. The teleportation locomotion system was mentioned to be very comfortable and easy to use. The simple one-button interaction model for accessibility to inexperienced users was also mentioned as a very good choice, easy to use, and comfortable. Finally, the experience and the scenarios were mentioned as thought-provoking and educational. Notably, several participants reported uncertainty or mild anxiety during dialogue choices, indicating genuine emotional engagement with the simulated scenario.

5 DISCUSSION AND FUTURE WORK

Web Application: The SUS analysis revealed an overall high usability score, confirming that participants found the web application functional. Notably, no participant provided comments suggesting usability barriers or confusion, which further reinforces the quantitative findings. However, as observed in the mean item analysis, a small number of responses suggest that some users might benefit from brief technical guidance or onboarding to become proficient more quickly.

VR Application: Overall, participants judged the prototype as usable and fit for purpose. The most important problems concern communicative clarity and pacing. An interesting insight emerged during post-experience discussions with participants. During the experience, most users expressed uncertainty or even mild anxiety, wondering if they had made the “right” choice at specific moments. Finally, another interesting finding from the VR experiment was that participants’ high level of independence occasionally resulted in avoidance behavior. Specifically, a few participants chose not to enter the area where the person of interest was located, effectively avoiding the main interaction sequence. This behavior is not necessarily negative, but it allows users to behave in accordance

with their own comfort levels and emotional reactions, reflecting the design's inherent realism and independence. Such freedom of movement and decision-making is a distinguishing feature of immersive virtual environments and can enhance the sense of presence. However, this outcome also highlights a potential trade-off between user autonomy and experiential completeness. By allowing full decision-making freedom, some users may unintentionally miss key parts of the narrative or the intended emotional learning objectives. This could reduce the overall effectiveness of the training scenario in promoting empathy and understanding, as critical experiential components might remain unexplored.

This thesis illustrates the potential of employing interactive technologies for early detection training in psychosis, yet it is constrained by specific limitations. First, the design of the system and implementation were centered on a single narrative environment, i.e., the home environment in which the user plays the role of the parent and the individual experiencing symptoms is the child. Moreover, the usability of the Web Application was constrained insofar as only a small number of participating clinicians were involved. In the same way, usability testing of the VR Application was held with only a limited number of users. Lastly, the project explicitly excludes therapeutic assessment and clinical trials. This thesis focuses on prototyping and design rather than clinical intervention. Consequently, while it demonstrates the potential and feasibility of VR-based training for early psychosis detection, it does not provide evidence of clinical efficacy or long-term impact.

While the current version of the system has demonstrated its potential to support training for early detection of psychosis through immersive VR experiences, several areas warrant further development and refinement. Real-world use issues that the quantitative analysis might have missed could also be better understood with the aid of a more detailed qualitative analysis combined with in-person interviews in both applications. At the same time, more participants would improve the the reliability and generalizability of results. Another interesting idea could be the expansion of the web platform. The ability to personalize the virtual setting in which the training is conducted is one way to enhance the capabilities of the web platform. Future developments of the system could focus on integrating artificial intelligence (AI) to make the training

scenarios more realistic and flexible. Finally, one interesting concept would be to enable the user to communicate freely with the NPC by utilising a real-time Speech-to-Text mechanism inside the VR application.

6 CONCLUSIONS

This thesis presented a system combining a web-based scenario authoring tool for clinicians with an immersive VR training application for relatives of individuals at risk of psychosis. The system enables non-specialists to experience clinically grounded, branching narrative simulations that model early psychosis symptoms through speech, thought, emotion, and embodied behavior. Evaluation with both clinical professionals (SUS: M = 81.4, "Excellent") and general users (CIQ: all factors above 3.8/5) demonstrated strong usability and engagement. The work provides a proof of concept for VR-based psychosis awareness training and points toward future clinical validation, expanded content, and AI-enhanced simulation.

REFERENCES

[1] Dimitrakopoulos, S., et al.: Don't blame psychosis, blame the lack of services: a message for early intervention from the Greek standard care model. BMC Psychiatry 22(1), 565 (2022).

[2] Treatment Advocacy Center: Research validates NAMI's Family-to-Family program. https://www.tac.org (2016). Accessed 20 Oct 2025.

[3] Tessier, A., et al.: Family psychoeducation to improve outcome in caregivers and patients with schizophrenia: a randomized clinical trial. Frontiers in Psychiatry 14, 1171661 (2023).

[4] Riches, S., et al.: Virtual reality-based training for mental health staff: a novel approach to increase empathy, compassion, and subjective understanding. Advances in Simulation 7(1), 19 (2022).

[5] Bridge, P., et al.: A virtual reality environment for supporting mental wellbeing of students on remote clinical placement: A multi-methods evaluation. Nurse Education Today 138, 106184 (2024).

[6] Spytska, L.: The use of virtual reality in the treatment of mental disorders such as phobias and PTSD. SSM – Mental Health 6, 100351 (2024).

[7] Chan, K.C., et al.: Application of Immersive Virtual Reality for Assessment and Intervention in Psychosis: A Systematic Review. Brain Sciences 13(3), 471 (2023).

[8] Yigitbas, E., et al.: VREUD – An End-User Development Tool to Simplify the Creation of Interactive VR Scenes. arXiv:2107.00377 (2021).

[9] Chen, M., Peljhan, M., Sra, M.: ConnectVR: A Trigger-Action Interface for Creating Agent-based Interactive VR Stories. In: IEEE VR 2024, pp. 286–297 (2024).

[10] Bangor, A., Kortum, P.T., Miller, J.T.: An empirical evaluation of the system usability scale. Int. J. Human–Computer Interaction 24(6), 574–594 (2008).

[11] Gao, M., Boehm-Davis, D.A.: Development of a customizable interactions questionnaire (CIQ) for evaluating interactions with objects in AR/VR. Virtual Reality 27(2), 699–716 (2023).

Cross-Scanner Breast Adenocarcinoma Image Segmentation with Deep Learning Algorithms

Dafni E. Tsolisou

ABSTRACT

Accurate segmentation of breast adenocarcinoma in hematoxylin and eosin (H&E)-stained whole slide images is critical for tumor grading, yet challenging due to domain shifts across imaging scanners. We present a domain-generalizable segmentation pipeline integrating DINOv2-Base, a vision foundation model, as the backbone of a modified DeepLabv3+ architecture. To address data scarcity and inter-scanner variability, we apply morphological, color, and HED-space stain augmentation, and evaluate training configurations via leave-one-scanner-out cross-validation. Our method achieves a cScore of 82.22%, the average of Dice and IoU, on the COSAS 2024 hidden test set spanning six scanners, three unseen during training, placing third among all submissions using modest computational resources.

Keywords: breast adenocarcinoma segmentation, semantic segmentation, vision foundation models, domain generalization, DINOv2, DeepLabv3+, digital pathology

ADVISOR

Emiris Ioannis, Professor, NKUA

1 INTRODUCTION

Breast cancer is the most prevalent malignancy among women worldwide and a leading cause of cancer-related mortality [1]. Early detection is critical, as evidenced by 2–4% annual mortality reductions in high-income countries with strong screening programs [2]. Accurate histopathological assessment of tumor grade from hematoxylin and eosin (H&E)-stained tissue sections is central to timely diagnosis and treatment planning, yet manual analysis by pathologists is labor-intensive and subject to inter-observer variability [3].

The digitization of stained tissue slides into whole slide images (WSIs) enabled digital pathology, facilitating remote diagnostics, archiving, and expert consultation [3]. Integrating digital pathology with deep learning (DL) gave rise to computational pathology (CPath), enabling automated detection, classification, and segmentation of disease patterns directly from WSIs [4], [5]. Among these tasks, semantic segmentation of cancerous tissue plays a crucial role in supporting pathologists during diagnosis and treatment planning.

A fundamental challenge in clinical deployment is domain shift: differences in scanner hardware, staining protocols, and sample preparation produce image appearance variations that violate the i.i.d. assumption of standard ML methods, causing performance drops when models are applied to new scanners [6], [7]. Addressing this in domain generalization (DG) setting, without access to target domain data during training, is essential for building robust models.

Recent self-supervised learning advances have produced vision foundation models (VFMs) that generate high-quality, task-agnostic features [8]. DINOv2 [9], a family of Vision Transformer (ViT)-based [10] models pre-trained on 142 million images, demonstrates strong zero-shot performance across diverse downstream tasks including segmentation.

Motivated by these developments and the Cross-Organ and Cross-Scanner Adenocarcinoma Segmentation (COSAS 2024) [11] challenge, this work investigates DINOv2 features for domain-invariant breast adenocarcinoma segmentation across scanners. The main contributions are:

1. A modified DeepLabv3+ [12] architecture using DINOv2-Base as backbone, adapted for dense prediction via a custom feature extraction module.
2. A systematic evaluation of encoder-decoder training strategies, including dual learning rate schedules with warm-up, via 3-fold leave-one-scanner-out cross-validation.
3. A comprehensive augmentation strategy combining morphological, photometric, and HED-space stain perturbations to promote scanner-invariant feature learning.
4. A competitive performance on the COSAS 2024 challenge dataset, achieving an 82.22% average Dice–IoU score on the hidden test set (3rd place equivalent), achieved using only moderate computational resources.

2 RELATED WORK

2.1 Deep Learning for Histopathological Segmentation

Deep learning methods have rapidly advanced the state of the art in computational pathology. Fully Convolutional Networks (FCNs) [13] introduced end-to-end semantic segmentation using a classic convolutional neural network (CNN) architecture with a final up-convolution step, while U-Net [14] improved spatial resolution recovery through encoder–decoder skip connections, becoming the gold standard in biomedical image segmentation. DeepLabv3+ [12] extended this paradigm by incorporating an atrous spatial pyramid pooling (ASPP) module to capture multi-scale context without resolution loss, achieving strong performance on general segmentation benchmarks.

For digital pathology specifically, the CAMELYON [15] dataset was created as one of the first benchmarks to compare DL methods performance in breast cancer metastasis detection. More recent models leverage self-supervised pre-training: CellVit [16] embeds ViT-based foundation models within U-Net for nuclei

segmentation, while MedSAM [17] fine-tunes the Segment Anything Model (SAM) [18] on 1.5 million medical image-mask pairs. These works collectively demonstrate the benefit of large-scale pre-training for data-scarce medical settings.

2.2 Domain Generalization in Histopathology

Domain shift in histopathology arises from scanner hardware differences, staining reagent variability, and tissue preparation protocols, producing distinct color distributions across imaging domains [6]. Standard mitigation strategies include stain normalization, domain adversarial training, and data augmentation [7]. Tellez et al. [7] showed that perturbing stain components directly in the hematoxylin-eosin-DAB (HED) color space is an effective augmentation strategy for simulating inter-scanner stain variability. Faryna et al. [19] further demonstrated that augmentation tailored to H&E imaging consistently outperforms generic augmentation for domain generalization in Cpath.

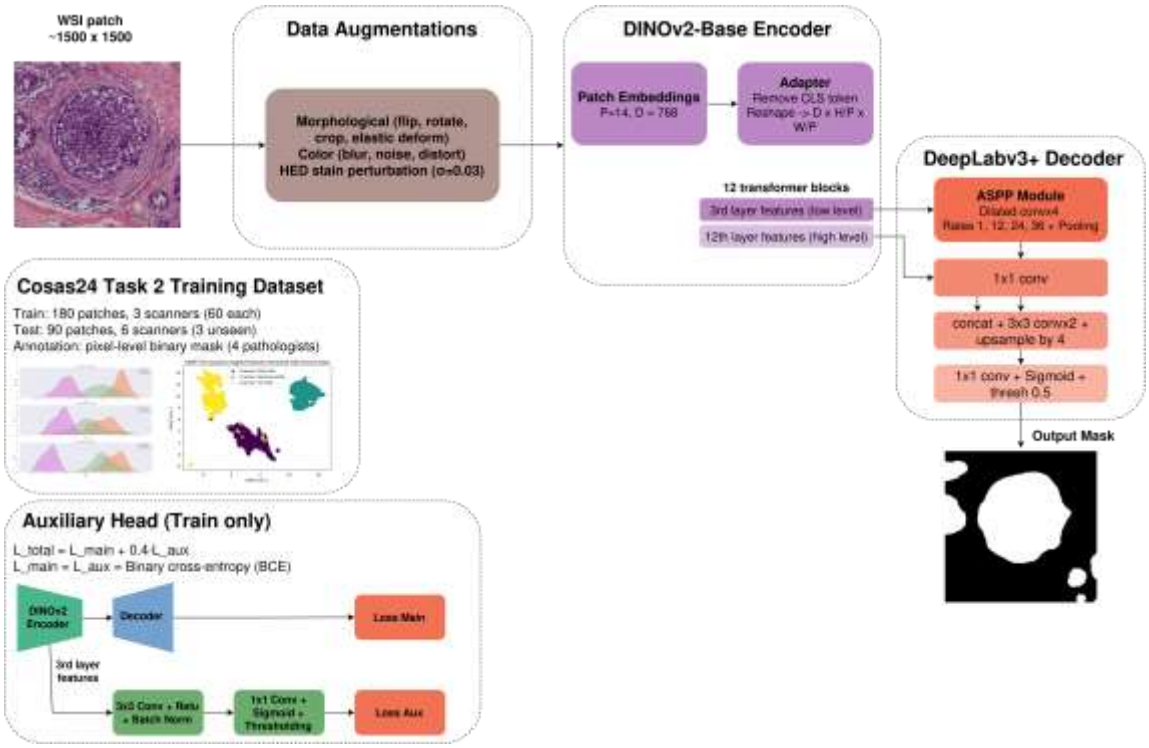


Fig. 1. Overview of the proposed pipeline. Dashed containers group the DINOv2-Base encoder and DeepLabv3+ decoder. Features from transformer blocks 3 and 12 feed the low-level and ASPP paths respectively. The auxiliary head is active only during training.

3 METHODS

3.1 DINOv2 Adapter and Backbone

DINOv2-Base (ViT-B/14) processes input images as sequences of 14×14 pixel patches. For an input of size H×W×C, the model produces patch embeddings of dimension D=768 after L=12 transformer blocks. Since semantic segmentation requires spatially dense predictions, a DINOv2 Adapter was implement that: (1) extracts the patch embedding matrix from selected transformer blocks, removing the CLS token, and (2) reshapes the resulting N_p×D sequence into a spatial feature map of dimensions D×(H/P)×(W/P), where P=14 is the patch size. This restores the spatial correspondence between features and input pixels, enabling subsequent convolutional processing. All training images were resized to 224×224 pixels before encoding.

3.2 Segmentation Architecture

The primary architecture (Fig. 1) follows the DeepLabv3+ encoder-decoder design with the DINOv2 Adapter replacing the conventional CNN backbone. Features from the 3rd transformer block serve as low-level encoder features, while features from the final (12th) transformer block feed the ASSP module. ASSP employs parallel atrous convolutions with dilation rates {1, 12, 24, 36} alongside global average pooling to capture multi-scale context. Afterwards, the decoder upsamples these features using bilinear interpolation and concatenates them with low-level feature. Subsequent steps of channel reduction, 3x3 convolution and a final upsampling produce a segmentation mask with the same size as the input image.

Following the PSPNet [20] paradigm, an auxiliary segmentation head was introduced (Fig. 1), that predicted a segmentation mask using features only from the 3rd transformer block. This head was trained alongside the main architecture, through an auxiliary loss, and provided an intermediate supervision signal that improved gradient flow to early layers. The auxiliary head was discarded at inference time.

Additionally, a lightweight baseline model was also evaluated in a linear probing setting. This used the DINOv2 Adapter output directly to train a simple classification head that predicted the final segmentation mask.

3.3 Learning Objective

Both models were trained with binary cross-entropy (BCE) loss. The total training loss combines the main decoder loss L_{main} and the auxiliary loss L_{aux} with a weighting factor:

$$L_{total} = L_{main} + 0.4 \cdot L_{aux} \quad (1)$$

An additional experiment was conducted that combined the BCE loss with the DICE loss to examine the effect of addressing class imbalance in the training mask.

$$L_{main} = L_{aux} = L_{BCE} + 0.3 \cdot L_{DICE} \quad (2)$$

3.4 Data Augmentation

In order to improve model generalization across scanners, on-the-fly image augmentation strategies were employed during training. These augmentations can be divided in two categories: morphological and color transformations. The first type of transformations were applied to both image and mask samples and include: random resize (±5%), random 224×224 crop, horizontal and vertical flip, random rotation (±90°), and elastic deformation. The second type include: Gaussian blur, Gaussian noise, and photometric distortion (brightness, contrast, saturation, hue adjustments), and a domain-specific augmentation strategy that perturbs stain components directly in the HED color space.

The latter was applied using a stain deconvolution method (Beer–Lambert law via scikit-image), that separates the H, E, and D channels and independently perturbed each one by inserting Gaussian noise (μ=1, σ=0.03). The channels were then recombined to produce a new RGB image with realistic simulated stain variation. This strategy, applied with probability 0.5, can effectively encode inductive bias against stain-based domain shift.

3.5 Training and Inference

Models were trained using the MMSegmentation framework with the AdamW optimizer. A linear warm-up was applied to the learning rate (LR) followed by polynomial decay. Various optimization policies with distinct encoder and decoder learning rates were evaluated. This dual learning rate strategy was selected in order to enhance model performance. A low encoder LR can prevent catastrophic forgetting of DINOv2 pre-trained features, while allowing the decoder to be adapted more freely with a separate LR.

During inference, WSI patches of 1500×1500 pixels are processed with a sliding window of size 224×224 and 30% overlap. Overlapping logit predictions were averaged before sigmoid thresholding at 0.5, mitigating boundary artifacts.

3.6 Dataset: COSAS 2024 Task 2

The COSAS 2024 Task 2 training set comprises 180 H&E-stained WSI patches (PNG, ~1500×1500 pixels, 20× magnification) of invasive breast adenocarcinoma of no special type, evenly distributed across three slide scanners: TEKSQRAY SQS-600P (teksqray-600p), KFBIO KF-PRO-400 (kfbio-400), and 3DHISTECH PANNORAMIC 1000 (3d-1000), with 60 patches per scanner. All WSIs originate from the Ruijin Hospital histopathology archive, making scanners the only source of domain shift. Ground truth binary masks were annotated at pixel level by four expert pathologists and reviewed by two additional experts, distinguishing normal glandular tissue (class 0) from adenocarcinoma (class 1). The hidden test set contains 90 patches from six scanners (15 each): the three training scanners plus three unseen scanners (leica-450, motic, 3d-250).

3.7 Evaluation Protocol

Model selection followed a 3-fold leave-one-out-scanner cross-validation on the training set approach. Performance was measured by the challenge composite score:

$$cScore = 0.5 \cdot mDice + 0.5 \cdot IoU \qquad (3)$$

where mDice and mIoU are the mean DICE coefficient and mean Intersection over Union computed across all pixels in the validation set. Final evaluation was performed on the hidden test set using the best model retrained on an 80-20 split of the full training set.

3.8 Computational Setup

All experiments were conducted on a server with four NVIDIA GeForce GTX 1080 GPUs (8GB VRAM each), a 12-core Intel Xeon E5-1650 v3 CPU at 3.50 GHz, and 32 GB system RAM. This constitutes a moderately resourced configuration.

4 RESULTS

4.1 Ablation Study

Table 1 summarizes 3-fold cross-validation results for all ablation configurations. The mean ± standard deviation (std) of cScore, mIoU, and mDice across the three folds are reported. The best-performing configuration (Policy 4) uses an encoder LR of 1×10^{-5}, decoder LR of 1×10^{-3}, and a 4,000-iteration warm-up followed by polynomial decay. This longer warm-up than the alternatives proved critical for stabilizing the pre-trained encoder weights.

Table 1. 3-fold cross-validation ablation results (mean± std across folds). DLv3+ = DeepLabv3+; FT = Finetuning; LR = Learning Rate.

Experiment	cScore(%)	mIoU(%)	mDice(%)	Backbone
Linear Probing	73.84 ± 4.22	67.30 ± 4.87	80.38 ± 3.56	Frozen
Baseline FT	75.32 ± 2.00	69.01 ± 2.33	81.63 ± 1.66	Full
DLv3+ Frozen	79.34 ± 5.11	73.84 ± 6.09	84.84 ± 4.14	Frozen
DLv3+ FT	75.89 ± 2.48	69.69 ± 2.89	82.09 ± 2.06	Full
DLv3+ Policy 1	80.65 ± 3.46	75.39 ± 4.18	85.91 ± 2.75	Dual LR

DLv3+ Policy 4	**82.02 ± 3.01**	**77.04 ± 3.66**	**86.99 ± 2.38**	**Dual LR**
DLv3+ 25% unfrozen	81.31 ± 6.00	76.24 ± 7.23	86.38 ± 4.77	Partial
DLv3+ combined loss	80.82 ± 3.33	75.60 ± 4.00	86.03 ± 2.65	Dual LR

Baseline experiments with frozen DINOv2-Base and a linear classifier achieved a cScore of 73.84±4.22 with linear probing, which improved to 75.32±2.00 when full fine-tuning was applied, confirming that DINOv2 zero-shot features are already informative for this task. Replacing the linear classifier with the DeepLabv3+ decoder significantly improved performance. Notably, a frozen backbone achieved 79.34±5.11, demonstrating the value of the ASPP module for capturing multi-scale pathological structures.

Naively fine-tuning both encoder and decoder simultaneously with a single learning rate degrades performance to 75.89±2.48, a result consistent with catastrophic forgetting. Introducing dual learning rates with a warm-up period progressively improved results, with Policy 4 achieving the best cScore of 82.02±3.01 and the lowest inter-fold variability. Partially fine-tuning only the final three transformer blocks (81.31±6) or combining BCE and Dice losses (80.82±3.33) yielded improvements over the full single-LR baseline but did not surpass Policy 4.

4.2 Final Model and Challenge Results

The final model, DeepLabv3+ with DINOv2-Base backbone trained under Policy 4 on an 80–20 split of the full training set, achieved a cScore of 82.22% on the hidden test set. Table 2 compares this result against the official COSAS 2024 Task 2 leaderboard.

Table 2. COSAS 2024 Task 2 final test phase leaderboard (top 5). † The proposed method was evaluated post-challenge and is not an official submission.

Team	cScore (%)	Rank
deep_microscopy	85.27	1st
Biototem	83.54	2nd
Daphne (Ours)	82.22	3rd†
Zhijian Life	81.92	4th
agaldran	81.75	5th

The proposed approach achieved third-place equivalent performance, outperforming the 4th and 5th ranked teams by 0.30 and 0.47 cScore points respectively. The top-ranked method (deep_microscopy, 85.27%) was separated from this work by 3.05 points, a gap attributable in part to the computational resource disparity.

Qualitative inspection of predicted masks (Fig. 2) revealed that the model accurately delineates large adenocarcinoma regions across all six test-set scanners, including the three unseen domains. Boundary imprecision is the primary failure mode: the model tends to overpredict near complex glandular borders in some scanners (kfbio-400, 3d-1000) and under-segments fine-grained cancerous regions in others (teksqray-600p, motic).

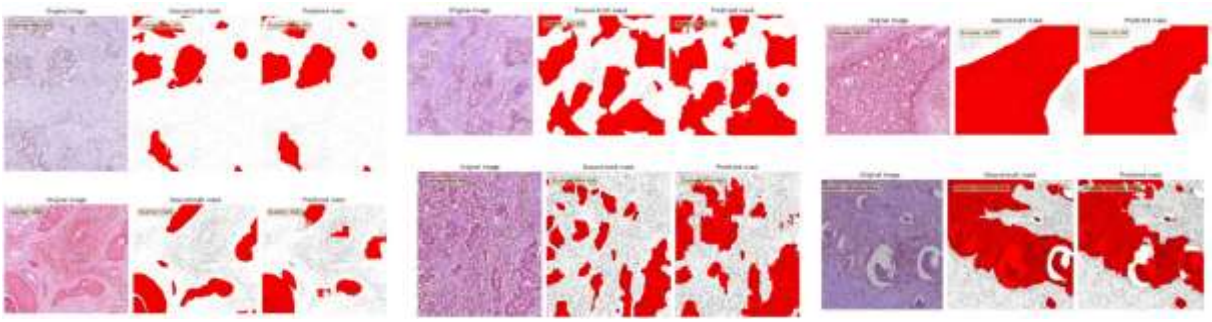


Fig. 2. Ground truth and predicted mask results of the final model on the test set. It is clear that the model struggles on the boundaries but can detect the adenocarcinoma regions effectively.

5 DISCUSSION

The results from this work demonstrate several important findings for cross-scanner segmentation in digital pathology. First, DINOv2-Base zero-shot features are significantly more informative for adenocarcinoma segmentation than randomly initialized encoders: even linear probing achieves 73.84% cScore. Second, the choice of decoder architecture matters more than full fine-tuning: the DeepLabv3+ decoder with a frozen backbone (79.34%) outperformed full fine-tuning with a single learning rate (75.89%), highlighting the importance of multi-scale context aggregation via ASPP.

Third, the dual learning rate strategy with a warm-up period was the single most impactful design choice: allowing the pre-trained encoder to adapt slowly while the decoder learned rapidly seemed to have acted against catastrophic forgetting and produced consistent inter-domain generalization. Fourth, HED-space stain augmentation contributed to robustness against the most distinct scanner (teksqray-600p), whose color distributions have minimal overlap with the other training scanners. Without this domain-specific perturbation strategy, performance on this scanner is expected to degrade more sharply.

The primary limitation of this study is the relatively small training set (180 patches from 3 scanners). While the selected cross-validation framework provided a valid estimate of generalization to unseen scanners, the small sample size posed a constrain, limiting the diversity of tissue morphologies and boundary conditions the model was exposed to during training. Additionally, the moderate computational resources prevented exploration of larger VFM variants (ViT-L/14, ViT-g/14) or ensemble methods. Future work should investigate these models and extend validation to additional cancer types and organs, given that the underlying pipeline is architecture-agnostic. Integration with segmentation-specific transformer decoders such as Mask2Former [21] or domain adaptation approaches such as LoRA [22] may further close the performance gap with the top challenge submissions.

6 CONCLUSION

This work presented a domain generalizable deep learning pipeline for semantic segmentation of breast adenocarcinoma across multiple WSI scanners. By integrating DINOv2-Base within a modified DeepLabv3+ architecture and employing comprehensive data augmentation, including HED-space stain perturbation, alongside a carefully designed dual learning rate training stratefy, the proposed method achieved a cScore of 82.22% on a six-scanner hidden test set, placing it third on the COSAS 2024 Task 2 leaderboard equivalent. Notably, these results are obtained using only moderate computational resources and 180 training patches, demonstrating the potential of adapting vision foundation models for specialized clinical imaging tasks. The contribution of this work is a lighweight, domain-aware framework with direct applicability to AI-assisted diagnostic tools in histopathology.

REFERENCES

[1] F. Bray et al., “Global cancer statistics 2022: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries,” CA. Cancer J. Clin., vol. 74, no. 3, pp. 229–263, 2024, doi: 10.3322/caac.21834.

[2] “Breast cancer.” Accessed: Mar. 19, 2026. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/breast-cancer

[3] B. Ehteshami Bejnordi et al., “Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Can-cer,” JAMA, vol. 318, no. 22, pp. 2199–2210, Dec. 2017, doi: 10.1001/jama.2017.14585.

[4] M. S. Hosseini et al., “Computational pathology: A survey review and the way forward,” J. Pathol. Inform., vol. 15, p. 100357, Dec. 2024, doi: 10.1016/j.jpi.2023.100357.

[5] A.Rahman et al., “Advances in tissue-based imaging: impact on oncology research and clinical practice,” Expert Rev. Mol. Diagn., vol. 20, no. 10, pp. 1027–1037, Oct. 2020, doi: 10.1080/14737159.2020.1770599.

[6] K. Stacke, G. Eilertsen, J. Unger, and C. Lundstrom, “Measuring Domain Shift for Deep Learning in Histopathology,” IEEE J. Biomed. Health Inform., vol. 25, no. 2, pp. 325–336, Feb. 2021, doi: 10.1109/JBHI.2020.3032060.

[7] D. Tellez et al., “Quantifying the effects of data augmentation and stain col-or normalization in convolutional neural networks for computational patholo-gy,” Med. Image Anal., vol. 58, p. 101544, Dec. 2019, doi: 10.1016/j.media.2019.101544.

[8] R. Bommasani et al., “On the Opportunities and Risks of Foundation Mod-els,” Jul. 12, 2022, arXiv: arXiv:2108.07258. doi: 10.48550/arXiv.2108.07258.

[9] M. Oquab et al., “DINOv2: Learning Robust Visual Features without Super-vision,” Feb. 02, 2024, arXiv: arXiv:2304.07193. doi: 10.48550/arXiv.2304.07193.

[10] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” ArXiv Prepr. ArXiv201011929, 2020.

[11] “Cross-Organ and Cross-Scanner Adenocarcinoma Segmentation - Grand Challenge,” grand-challenge.org. Accessed: Mar. 19, 2026. [Online]. Available: https://cosas.grand-challenge.org/cosas/

[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation,” Aug. 22, 2018, arXiv: arXiv:1802.02611. doi: 10.48550/arXiv.1802.02611.

[13] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” Mar. 08, 2015, arXiv: arXiv:1411.4038. doi: 10.48550/arXiv.1411.4038.

[14] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Net-works for Biomedical Image Segmentation,” May 18, 2015, arXiv: arXiv:1505.04597. doi: 10.48550/arXiv.1505.04597.[15]G. Litjens et al., “1399 H&E-stained sentinel lymph node sections of breast cancer patients: the CAMELYON

dataset," GigaScience, vol. 7, no. 6, p. giy065, May 2018, doi: 10.1093/gigascience/giy065.

[15] F. Hörst et al., "CellViT: Vision Transformers for precise cell segmenta-tion and classification," Med. Image Anal., vol. 94, p. 103143, May 2024, doi: 10.1016/j.media.2024.103143.

[16] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, "Segment anything in medical images," Nat. Commun., vol. 15, no. 1, p. 654, Jan. 2024, doi: 10.1038/s41467-024-44824-z.

[17] A. Kirillov et al., "Segment Anything," Apr. 05, 2023, arXiv: arXiv:2304.02643. doi: 10.48550/arXiv.2304.02643.

[18] K. Faryna, J. van der Laak, and G. Litjens, "Tailoring automated data augmentation to H&E-stained histopathology," in Proceedings of the Fourth Conference on Medical Imaging with Deep Learning, PMLR, Aug. 2021, pp. 168–178.

[19] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Net-work," Apr. 27, 2017, arXiv: arXiv:1612.01105. doi: 10.48550/arXiv.1612.01105.

[20] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention Mask Transformer for Universal Image Segmentation," Jun. 15, 2022, arXiv: arXiv:2112.01527. doi: 10.48550/arXiv.2112.01527.

[21] E. J. Hu et al., "Lora: Low-rank adaptation of large language models.," ICLR, vol. 1, no. 2, p. 3, 2022.

metagRoot: A Comprehensive Database of Protein Families Associated with Plant Root Microbiomes

Maria N. Chasapi

ABSTRACT

The plant root microbiome is vital in plant health, nutrient uptake, and environmental resilience. To explore and harness this diversity, we present metagRoot, a specialized and enriched database focused on the protein families of the plant root microbiome. MetagRoot integrates metagenomic, metatranscriptomic, and reference genome-derived protein data to characterize 71,091 enriched protein families, each containing at least 100 sequences. These families are annotated with multiple sequence alignments, CRISPR elements, Hidden Markov Models, taxonomic and functional classifications, ecosystem and geolocation metadata, and predicted 3D structures using AlphaFold2. MetagRoot is a powerful tool for decoding the molecular landscape of root-associated microbial communities and advancing microbiome-informed agricultural practices by enriching protein family information with ecological and structural context. The database is available at https://pavlopoulos-lab.org/metagroot/ or https://www.metagroot.org

ADVISORS

Ioannis Emiris, Professor, NKUA

Dimitrios Stravopodis, Associate Professor, NKUA

Georgios Pavlopoulos, Director of Research, BSRC “Alexander Fleming”

1 INTRODUCTION

The discovery of the roles of proteins whose expression takes place in distinct plant microenvironments, such as aerial roots, bulbs, endosphere, nodules, rhizoplane, rhizosphere, stems, and stem tubers, is crucial for the continuous progress of plant biology. These compartments are the hosts of dissimilar groups of proteins associated with nutrient transport, signal responses, plant-microbe stress reactions, and vegetation-plant interactions. Instead of being evenly distributed throughout the plant, most of these proteins are spatially restricted, pointing to each environment subtype's unique physiological and ecological roles. For example, proteins involved in nitrogen metabolism are primarily located in nodules. However, those associated with reactive oxygen species signaling, redox regulation, and cell wall remodeling are mainly found in rhizoplane and rhizosphere, where a plant is exposed to soil microbes and abiotic stressors (1).

Plant compartments provide a biological habitat for diverse microorganisms (bacteria, fungi, archaea) closely linked with plant proteins to govern growth, immunity, and metabolism. For example, the secretion of host proteins, including expansins, defensins, and peroxidases, can alter the microbial composition and activity in the endospheres and rhizoplane. As a result, these plant proteins' expression and activity can be changed by microbial metabolites, generating a well-regulated system and a great deal of dynamics. These relationships are not merely beneficial but often essential. Symbiotic microbes in nodules, for example, provide fixed nitrogen, while root-associated microbiota can enhance phosphate solubilization, stress tolerance, and pathogen resistance through coordinated interactions with plant proteins (2).

Generic protein and protein family databases such as UniProtKB, Protein Data Bank (PDB), RefSeq, GenBank, IMG/M, Uniparc, Big Fantastic Database (BFD), MGnify, Pfam, InterPro, SMART, NMPFamsDB, COGs and others provide curated information on protein function, structural information, and interactions for most reference and metagenomic proteomes. Meanwhile, emerging resources like CyanoBase and RhizoBase, as well as the Microbial Genomes Atlas (MiGA), facilitate cross-referencing between plant proteomic data and the taxonomic and

functional profiles of associated microbes. However, to our knowledge, no existing database integrates reference proteomes with metagenomic and metatranscriptomic proteins at the protein family level for plant roots and their subhabitats. By examining proteins and protein families unique to specific plant environments or shared across certain subhabitats, researchers can delve deeper into the functional diversity underpinning plant adaptation and health, potentially uncovering new molecular functions and biological processes. In light of this, we introduce metagRoot, a comprehensive catalog of microbial protein families specific to plant root compartments (2–4).

2 MATERIALS AND METHODS

2.1 Data collection, filtering, and protein family generation

In October 2023, all publicly accessible plant root-associated metagenomes, metatranscriptomes, and reference genomes were sourced from the Integrated Microbial Genomes & Microbiomes (IMG/M) database (5), maintained by the Joint Genome Institute. The compiled dataset included 1,280 assembled metagenomes and 335 metatranscriptomes, collectively containing 3,497,616,101 protein-coding sequences and 3,207,863,396 scaffolds. Filtering criteria were applied to ensure a dataset of high-quality protein predictions (Figure 1) (6). To reduce truncation risk, sequences within 10 nucleotides of scaffold ends were removed. Additionally, only scaffolds longer than 500 nucleotides were retained to ensure that mainly complete genes were represented. Low-complexity regions were masked using the tantan application (version 26) to further refine the dataset. Any remaining sequences shorter than 35 amino acids were discarded. From this dataset, only samples linked to aerial roots, nodules, rhizoplane, endosphere, rhizosphere, stems, and stem tubers according to GOLD classification ecosystem subtypes (7), comprising 1,527 datasets (1,199 metagenomes, 327 metatranscriptomes), with 29,019,966 scaffolds and 403,495,896 proteins. For reference genomes, 17,825,681 protein sequences were extracted from 2,978 bacterial isolate genomes and 2,428 sequences from one root-associated archaeal isolate genome, yielding 2,979 datasets and a total of

421,324,005 sequences in the final catalog. For protein family generation, the MMseqs2 Linclust clustering algorithm was executed in bidirectional mode with parameters set to 30% sequence identity and 80% coverage. Protein families with 100 or more members were selected and aligned using MAFFT. During alignment, trimming was applied to minimize gaps. After aligning the sequences, the hhfilter was employed to reduce redundancy within each family by applying a 95% sequence identity threshold and a 70% coverage criterion. This step minimizes bias from highly similar sequences and improves the accuracy of subsequent structural predictions. To preserve biodiversity and ensure rich informational content, only families that originally contained 100 or more members before applying the hhfilter were reported. The final dataset consists of 71,091 families from 4,480 datasets, 25,961,294 (metagenome) scaffolds, 3,070,323 reference genomes, and 38,700,093 proteins.

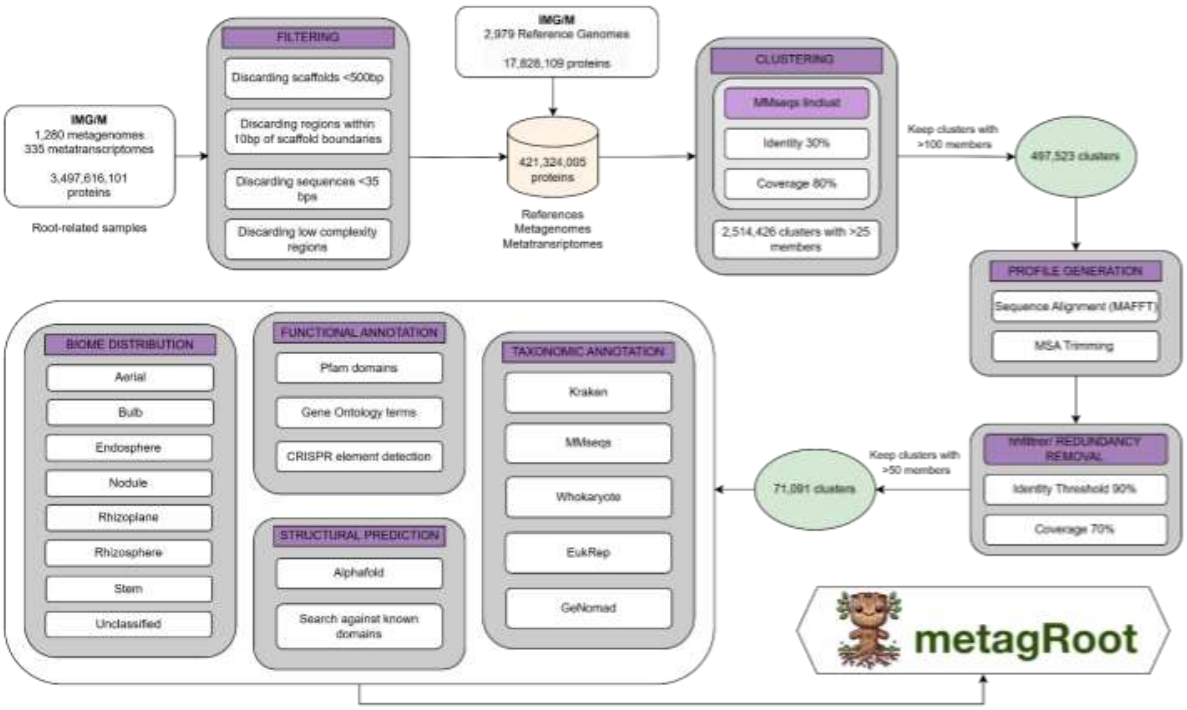


Figure 1. Data preprocessing and filtering. Pipeline illustrating the sequence filtering and clustering methodology for generating protein families from both reference genomes and metagenomes/metatranscriptomes.

2.2 Taxonomy

The taxonomic analysis focused on protein families with more than 100 members to ensure statistical robustness. The scaffold sequences corresponding to protein families were retrieved from the IMG/M database. Classification began with Kraken2, known for its high accuracy, and was further refined by applying MMseqs taxonomy to annotate eukaryotic proteins better and reduce false positives. For scaffolds that remained unclassified, Whokaryote was used to differentiate eukaryotic from prokaryotic contigs based on domain-specific gene structural features, followed by EukRep to detect eukaryotic sequences further. Additionally, GeNomad was employed to identify plasmid and viral sequences, ensuring a comprehensive assignment of mobile genetic elements. Ultimately, this multi-tool approach resulted in 37,155,632 scaffolds classified as Bacteria, 63,417 as Archaea, 45,286 as Eukaryota, 15,844 as Viruses, 6,582 identified as plasmid-derived, two linked to organelles, one classified as synthetic, and 1,387,727 remaining unclassified.

2.3 Biome and metadata collection

Insights into habitat-specific diversity were obtained by examining the distribution of these protein families across various ecosystems, including aerial roots, bulbs, nodules, the rhizoplane, endosphere, rhizosphere, stems, and stem tubers, as defined by the GOLD classification. Environmental metadata were extracted from the IMG/M database and organized using the GOLD taxonomy, which categorizes ecosystems hierarchically into Environmental, Engineered, and Host-Associated groups. Within the Host-Associated microbiome category, a focus was placed on root-associated ecosystem subtypes. Within the Host-Associated microbiome category, root-associated ecosystem subtypes were prioritized, with dataset distribution before taxonomic family classification showing 1,898 rhizosphere entries, followed by 1,209 unclassified root-associated systems, 970 nodule samples, 199 rhizoplane datasets, 188 endosphere records, 29 aerial root collections, 10 stem tuber examples, and one dataset each documenting bulbs and stems. After family generation, among the 71,091 families containing 100 or more members, the composition was slightly adjusted to 1,879 rhizosphere datasets, 1,205 unclassified root-associated datasets, 970

nodule datasets, 199 rhizoplane datasets, and 186 endosphere datasets. Additionally, fewer datasets remained from other ecosystem subtypes, including 29 aerial root samples, 10 stem tuber samples, and one sample each from bulbs and stems.

2.4 Functional annotation and CRISPR elements

Protein families were annotated by searching the representative sequences against Pfam (v37, Dec 2024), which includes 21,979 families and 709 clans, using hmmsearch (HMMER 3.3.2) with the trusted cutoff. In total, 66,854 families were successfully annotated with Pfam hits. CRISPR-related families were identified by first detecting CRISPR arrays using CRT (CRISPR Recognition Tool), which recognizes direct repeats and spacers. In the total dataset of 25,961,294 scaffolds from metagenomes and metatranscriptomes in the 71,091 families, 8,927 were found with CRISPR arrays. Additionally, the protein families were screened for CRISPR-associated (Cas) proteins and other functional components based on Pfam hits. To this end, 1,223 scaffolds were detected with CRISPR Cas elements. At a family level, 24,078 of them were found to contain CRISPR arrays, whereas 6 families were annotated for Cas systems based on the CRISPR-associated Pfam domains. Collectively, 6 families were found to have a complete CAS system (both arrays and Cas elements).

2.5 Structural model prediction

Structural information serves as a critical complement to sequence analysis, either confirming predicted functional elements by revealing their three-dimensional organization or refining annotations that sequence data alone cannot resolve. To this end, AlphaFold2 was used to predict structures for protein families with at least 50 effective sequences (upon hhfiltering for redundancy removal). For each family, five models were generated, and the one with the highest pLDDT score was selected. Only models classified as high (pTM ≥0.7) or medium-quality (pTM between 0.5 and 0.7) were retained. These models were then compared to experimentally validated structures from CATH (v4.4, Feb 2023) and PDB (2024 release), as well as both experimental and predicted structures in AlphaFoldDB, using Foldseek. Matches with a TM-score greater than 0.5 were

considered valid. For families that lacked TM-alignment hits, additional criteria were applied to ensure alignment relevance. If the query was shorter than the target, the query alignment score had to be above 0.5. Similarly, if the query was longer, at least 50% of the target had to be aligned. Finally, superfamilies were defined using Foldseek in bidirectional mode, grouping structures based on 80% coverage and a TM-score of at least 0.5. The final dataset consisted of 71,088 family structural models, with 59,059 classified as High Quality and 9,036 as Medium Quality, and subsequently grouped into 9,463 superfamilies. Comparative results and structural analyses with CATH, PDB, and AlphaFold revealed a total of 66,837 hits in CATH, including four unique entries; 66,910 hits in PDB, with no unique entries; and 68,049 hits in AlphaFold, with 624 unique entries. The analysis led to the discovery of 28 novel structures.

2.6 Database Implementation and Structure

The metagRoot database uses MySQL as its main database management system and is built on the ASP.NET Core MVC framework. The back-end is coded in C#, while the front-end employs HTML, CSS, and JavaScript. MetagRoot also utilized R for statistical calculations and data analysis, all within the Model-View-Controller (MVC) pattern. The system is divided into Models, Views, Controllers, Services, and Factories. Models, represented by C# classes, define data and apply validation rules. Views use Razor (.cshtml) templates and C# logic to format data and generate HTML. Controllers act as intermediaries between Models and Views. Services manage core application logic and data operations, whereas Factories create specific components to minimize dependencies. When a user sends a request, the Controller identifies the required action, uses a Factory to generate the necessary Model, and retrieves data through Services. Once the Controller has organized the data, it passes everything to the View for HTML rendering, thereby completing the request-response cycle.

Finally, the application comes with a suite of integrated plugins and viewers. The integrated components include Diamond, which is utilized for rapid sequence alignment; the nextProt Feature Viewer, employed for the visualization of annotated sequence features; UpSet plots for showing intersections and unions, MSAViewer, used to display multiple sequence alignments; the SkyLign sequence

logo viewer for visualizing HMM models and frequencies, and the Mol* structure viewer, implemented for interactive 3D molecular structure visualization. Additionally, OpenStreetMap is incorporated to support geospatial data representation, and HMMER is used for HMM-based sequence homology searches.

3 RESULTS

3.1 Database components

MetagRoot is a specialized repository cataloging 71,091 protein families linked to plant root microbiomes, each comprising ≥100 sequences derived from integrated metagenomic (1,199 datasets), metatranscriptomic (327 datasets), and 2,978 reference genomes. These families analyzed across 29 million scaffolds and 421 million protein sequences are enriched with annotations including multiple sequence alignments, Hidden Markov Models (HMMs), taxonomic/functional classifications, CRISPR elements, and ecosystem metadata (e.g., rhizosphere, nodules). Structural insights are provided via AlphaFold2-predicted 3D models, with 59,059 high-quality and 9,036 medium-quality structures.

3.2 The metagRoot Interface

The metagRoot platform is accessed via a top-of-page navigation bar that provides main menus such as Browse, Sequence Search Tools, Statistics, and Downloads, ensuring access to every feature from any page (Figure 2).

3.3 Browse Families and Search Filters

Under Browse → Families, users encounter a comprehensive query interface that supports both simple and advanced searches. A free-text box accepts metagRoot family identifiers or keywords linked to family identifiers, while dropdown menus allow filtering by sequence category (Metagenome-only, Metatranscriptome-only, Mixed, or all families), taxonomy, and sequence type (e.g., CRISPR).

Ecosystem checkboxes (Aerial Root, Bulb, Endosphere, Nodule, Rhizoplane, Rhizosphere, Stem, Stem Tuber, Unclassified) further narrow results, and a minimum-members slider is used for the retrieval of families with at least a percentage of their members coming from a specific habitat. The search options are further combined with an AND/OR logic toggle. Finally, users can constrain families by size before initiating the search. Once the criteria are set, a prominent search button retrieves matching families for online browsing or export.

3.4 Family Entry: Data, Visualization, and Metadata Context

When a family is selected, the interface presents its unique identifier along with key summary statistics such as the total number of member sequences, contributing Plant Datasets, scaffolds, and the average sequence length. The representative sequence from the multiple alignment is displayed in full (complete with its amino-acid string and length) followed by comprehensive tables listing every associated Plant Dataset (including dataset ID, environmental metadata, and designation as metagenome or metatranscriptome) and every scaffold (showing scaffold ID, source dataset linkage, taxonomic assignment, and sequence length). All identifiers are linked to IMG/M.

Functional annotation is provided through Pfam domain hits, each annotated with start and end positions and per-domain confidence scores. Structural annotation reports the top five PDB, CATH, or AlphaFoldDB matches for the family's representative sequence, listing TM-scores and alignment spans to convey fold similarity.

In addition, a suite of interactive tools enables in-depth exploration of these data. The multiple sequence alignment (MSA) can be viewed in "full" or "seed" mode, with options to apply custom coloring schemes, set identity and occupancy thresholds, search by motif or regular expression, and download the alignment in FASTA format. An adjacent sequence logo viewer renders the family's profile HMM as an interactive plot of residue probabilities (each position being clickable to reveal detailed scores). The underlying HMM can be exported in HMMER or HH-suite format. Per-residue predictions for transmembrane topology, secondary structure elements, conservation histograms, and Pfam (or other)

domain annotations are mapped directly onto the representative sequence in a linear feature viewer. An AlphaFold2 model completes the picture when available by providing a rotatable, three-dimensional ribbon diagram of the predicted structure.

To contextualize the family ecologically and taxonomically, metagRoot integrates several metadata summaries. Phylogenetic assignments aggregated across all scaffolds reveal the family's higher-level taxonomic composition, while a gene-neighborhood overview highlights conserved Pfam domains in the genomic vicinity of each family member. Ecosystem distributions are quantified in both a table and a pie chart, showing the percentage of sequences derived from each habitat category. A second pie chart breaks down members by data type (metagenome, metatranscriptome, isolate), and a world map plots the geographic coordinates of all sampling sites. Finally, a bar chart summarizes the family's taxonomic breadth at the phylum or class level.

3.5 Data Accessibility & Export

Every alignment, profile, annotation table, visualization, and structural model within metagRoot is fully exportable. Sequence alignments and family HMMs can be downloaded in standard FASTA and HMMER/HH-suite formats, respectively. The three-dimensional models are available in PDB format, and metadata tables may be exported as CSV or TSV files.

3.6 Sequence Search & Visualization Tools

Beyond browsing families, the Sequence Search Tools menu provides integrated HMMER and DIAMOND pipelines for querying the entire metagRoot database with user-supplied sequences or profile hidden Markov models. Users can refine search results by taxonomy or ecosystem, adjust sensitivity parameters such as cut-offs, substitution matrices, and gap costs, and then download tabulated hits in plain text format. A dedicated Pattern Search interface supports PROSITE-style motif queries or arbitrary regular expressions, enabling rapid detection of conserved sequence features across all family members.

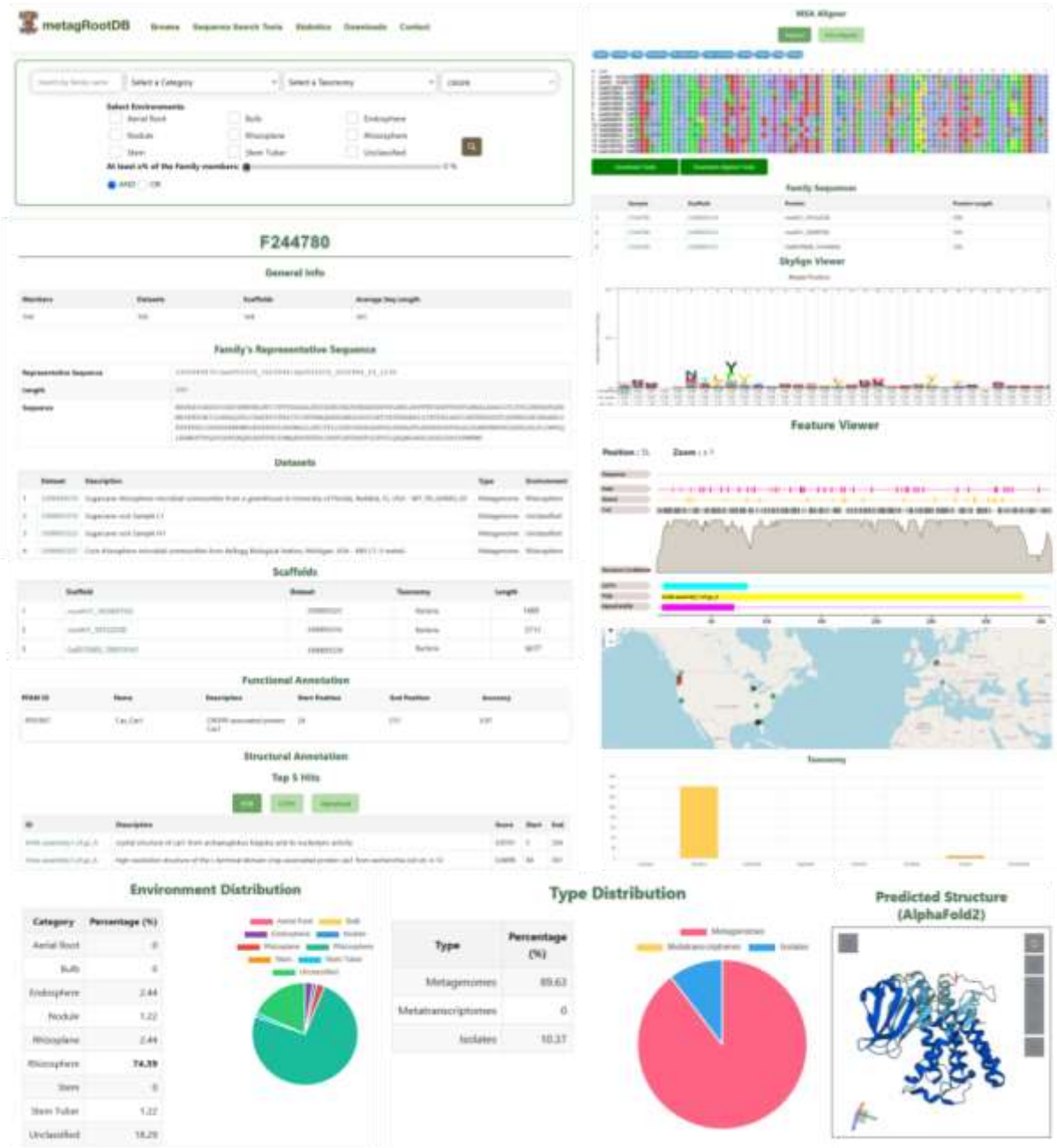


Figure 2. MetagRoot's interface and summaries. The Browse Families page offers free-text search and filters for sequence category, taxonomy, ecosystem, and family size. Selecting a family displays its identifier, member, and dataset counts, average sequence length, and representative amino-acid sequence, followed by tables of associated Plant Datasets and scaffolds, Pfam functional domains, and top structural matches. Interactive viewers include a multiple sequence alignment with customization and FASTA export, a sequence logo from the profile HMM with export options, a linear feature map of topology, secondary structure, conservation, and domain annotations, and an embedded AlphaFold2 3D model. Metadata panels summarize phylogenetic assignments, gene-neighborhood contexts, ecosystem and data-type distributions, and global sampling locations. All alignments, profiles, models, and tables are downloadable in standard bioinformatics formats.

4 CONCLUSION

The properties of the protein families identified in this study were initially examined for several patterns (Figure 4A). In the near-normal distribution of protein lengths, most proteins spanned 200 to 450 aminoacids, whereas a clear peak was observed between the range of 250 and 300. In addition, protein family sizes were skewed towards smaller groups, as many contained, on average, 150 to 300 members. In the case of metagenomes and metatranscriptomes, scaffold lengths were found to lie predominantly between the range of 1,000 and 5,000 base pairs. Lastly, the histogram illustrating the protein members per cluster underscored a tendency for these proteins to cluster mainly between 150 and 500 sequences.

In terms of taxonomy (Figure 4B), bacterial taxa were determined to constitute the vast majority (approximately 94.91%) of all classified sequences. Archaeal sequences, fewer in number, accounted for a minority (0.21%). Conversely, eukaryotic and viral taxa, as well as organelles, plasmids, and synthetic categories, represented only a small fraction of the dataset (0.21%), whereas 4.67% remained unclassified. This taxonomic distribution underscores the bacterial-centric nature of the microbial communities represented while emphasizing that archaea, though present, comprise only a small proportion relative to bacteria.

The overlap among metagenomes, metatranscriptomes, and reference genomes is shown in a Venn diagram (Figure 4C). A substantial core consisting of 24,712 sequences was shared across all three categories. Pairwise intersections were also significant, notably between metagenomes and reference genomes (22,969 sequences), as well as between metagenomes and metatranscriptomes (10,404 sequences). Metatranscriptomes and reference genomes, however, exhibited minimal exclusivity, containing only four and five unique sequences, respectively. Thus, metagenomic datasets effectively encompass most sequence diversity, capturing sequences found within reference genomes and metatranscriptomic datasets. Collectively, these results depict an ecosystem dominated by bacterial sequences characterized by predominantly small protein clusters across certain habitats.

Lastly, the biome distribution of protein families illustrated in the heatmap revealed significant variability across different plant-associated habitats (Figure 4D). One noteworthy observation was the prevalence of protein families found only in certain biomes (Rhizosphere and Nodule), suggesting unique functional adaptations. The rhizosphere accounted for the largest share of these exclusive families, aligning with its dynamic role in plant-microbe relationships. Likewise, a set of families (>25,000 families) is shared between the endosphere, rhizosphere, and rhizoplane, hinting at the specialized demands within internal tissues and along root surfaces, and reflecting shared functions tied to root interactions and resource uptake. In contrast, habitats like bulbs and nodules demonstrated fewer overlaps, maybe due to their unique physiology and characteristics. Bulbs, unlike the other environments discussed, are storage organs characterized by their layered structure, high sugar content, and low oxygen levels. Similarly, nodules are formed due to symbiotic interaction between the plant and nitrogen-fixing bacteria, and as a result, they provide an environment favorable for nitrogenase (8).

In conclusion, metagRoot represents a significant advancement in the study of plant root microbiomes by integrating metagenomic, metatranscriptomic, and reference genome data into a unified, richly annotated resource. By cataloging 71,091 protein families with structural, functional, and ecological context, metagRoot bridges a critical gap in existing databases, enabling researchers to explore the molecular mechanisms underpinning plant-microbe interactions with unprecedented depth. The inclusion of AlphaFold2-predicted 3D models, CRISPR elements, habitat-specific metadata, and interactive analytical tools empowers users to uncover novel protein functions, refine annotations, and investigate evolutionary and ecological dynamics across root-associated niches. As a publicly accessible platform, metagRoot not only accelerates hypothesis-driven research but also fosters microbiome-informed innovations in agriculture.

REFERENCES

[1] Charpentier,M. and Oldroyd,G.E.D. (2013) Nuclear calcium signaling in plants. Plant Physiol, 163, 496–503.

[2] Vannier,N., Mony,C., Bittebière,A.-K. and Vandenkoornhuyse,P. (2015) Epigenetic Mechanisms and Microbiota as a Toolbox for Plant Phenotypic Adjustment to Environment. Front Plant Sci, 6, 1159.

[3] Faure,D., Bonin,P., Duran,R., and Microbial Ecology EC2CO consortium (2015) Environmental microbiology as a mosaic of explored ecosystems and issues. Environ Sci Pollut Res Int, 22, 13577–13598.

[4] Bulgarelli,D., Schlaeppi,K., Spaepen,S., Ver Loren van Themaat,E. and Schulze-Lefert,P. (2013) Structure and functions of the bacterial microbiota of plants. Annu Rev Plant Biol, 64, 807–838.

[5] Chen,I.-M.A., Chu,K., Palaniappan,K., Ratner,A., Huang,J., Huntemann,M., Hajek,P., Ritter,S.J., Webb,C., Wu,D., et al. (2023) The IMG/M data management and analysis system v.7: content updates and new features. Nucleic Acids Research, 51, D723–D732.

[6] Pavlopoulos,G.A., Baltoumas,F.A., Liu,S., Selvitopi,O., Camargo,A.P., Nayfach,S., Azad,A., Roux,S., Call,L., Ivanova,N.N., et al. (2023) Unraveling the functional dark matter through global metagenomics. Nature, 622, 594–602.

[7] Mukherjee,S., Stamatis,D., Bertsch,J., Ovchinnikova,G., Sundaramurthi,J.C., Lee,J., Kandimalla,M., Chen,I.-M.A., Kyrpides,N.C. and Reddy,T.B.K. (2021) Genomes OnLine Database (GOLD) v.8: overview and updates. Nucleic Acids Research, 49, D723–D733.

[8] Maróti,G. and Kondorosi,E. (2014) Nitrogen-fixing Rhizobium-legume symbiosis: are polyploidy and host peptide-governed symbiont differentiation general principles of endosymbiosis? Front Microbiol, 5, 326.