

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря СІКОРСЬКОГО»  
Навчально-науковий Фізико-технічний інститут

## РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з кредитного модуля «Інтелектуальні обчислення»

на тему:

**"РОЗПІЗНАВАННЯ РОБОЧОЇ ПОВЕРХНІ СТОЛУ ЗД ПРИНТЕРА ПРЯМИМИ ТА  
ІТЕРАЦІЙНИМИ МЕТОДАМИ, НЕЙРОННИМИ МЕРЕЖАМИ"**

Виконав:

студент 3 курсу НН ФТІ  
Голуб Михайло Вікторович  
номер залікової книжки \_\_\_\_\_

Перевірив: \_\_\_\_\_

Оцінка: \_\_\_\_\_

Київ– 2025

## Зміст

# 1. Вступ

## 1.1 Актуальність

3D принтери технології пошарового наплавлення філаменту (Fused filament fabrication, далі – FFF 3D принтери) зараз є найбільш поширеними верстатаами для швидкого прототипування. Данна технологія має недолік високої кількості браку. Збільшення кількості завчасних зупинок друку через брак може бути досягнуто при використанні методів що використовують контекст ( положення друкованого об'єкту в просторі відносно камери), аніж при використанні нейронних мереж які аналізують зображення без жодної додаткової інформації про 3D модель що друкується. Для розробки і застосування покращених методів необхідно чітко знати положення столу, щоб визначити де на фотографії / відео має знаходитись 3D модель, яка друкується. Першим кроком з створення таких методів є система розпізнавання робочої поверхні столу FFF 3D принтера.

## 1.2 Мета роботи

Мета роботи – побудувати і порівняти методи розпізнавання робочої поверхні столу для FFF 3D принтера Bambulab A1 mini.



Рис. 1.1: Bambulab A1 mini

## 2. Допрограмний етап

### 2.1 Аналіз об'єкту, що розпізнається

Необхідно розпізнати робочу поверхню столу FFF 3D принтера Bambulab A1 mini. Дано модель має декілька різних змінних робочих поверхонь, але найбільш розповсюджена – PEI-пластина.

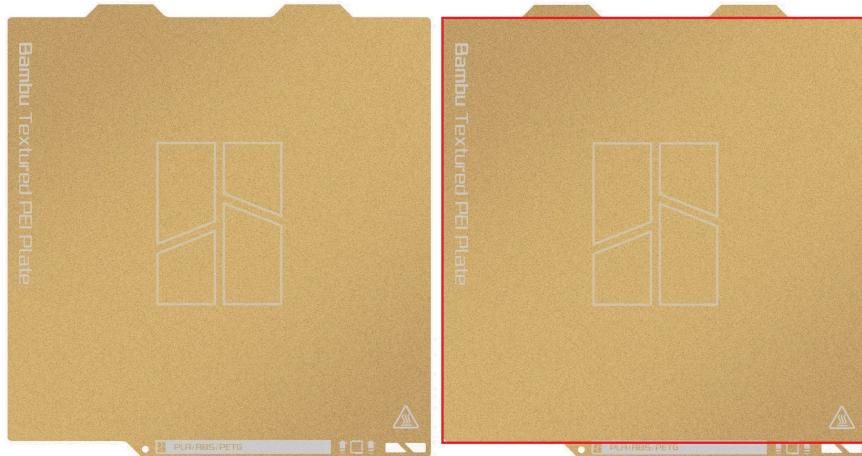


Рис. 2.1: PEI-пластина. На зображенні справа показано робочу зону

Робоча область PEI-пластини 18x18 см. Колір майже всіх PEI-пластин бронзово-золотистий. Отже, можна спробувати створити методи що шукають на зображенні проекції квадратів, конкретний колір, або проекції і колір одночасно. Також, при використанні кольору як критерію пошуку, можна скористатись тим, що корпус даного 3D принтера в більшості місць білий.

Окрім роботи з значеннями пікселів для визначення кольору, можна скористатись тим, що робоча поверхня дуже контрастна з оточенням, корпусом принтера, і застосувати фільтр границь.

### 2.2 Постановка задачі

Програма має для вхідного зображення повернути 8 значень – по дві координати чотирьох кутів столу/робочої області.

### 2.3 Обрання методів розв'язку задачі

Задачу розпізнавання об'єкту можуть виконати нейронні мережі, за наявності достатньої кількості розміщених зображень і функції втрат. Окрім нейронних мереж, можна застосувати ройові алгоритми, за умови наявності фітнес функції. Також, можна створити додатковий контекст, який гарантовано враховуватиме границі об'єктів та їх колір. Такий контекст можна подавати на вхід нейронних мереж чи ройових алгоритмів.

### 2.4 Створення методів отримання контексту

#### 2.4.1. Отримання контексту з кольору пікселів

З кольору пікселів можна створити контекст ввівши ідеальне значення кольору і допустимі відхилення. Доцільно це робити використавши HSV (трьохканальна система кольорів "Колірний тон, насиченість, яскравість"), а не RGB (трьохканальна система кольорів "Червоний, зелений, синій"),

оскільки в HSV можна проігнорувати яскравість та враховувати лише колірний тон і насиченість. Щоб отримати ідеальне значення кольорів столу і корпусу використано фотографію принтеру в типових умовах освітлення, отриману на камеру телефона.

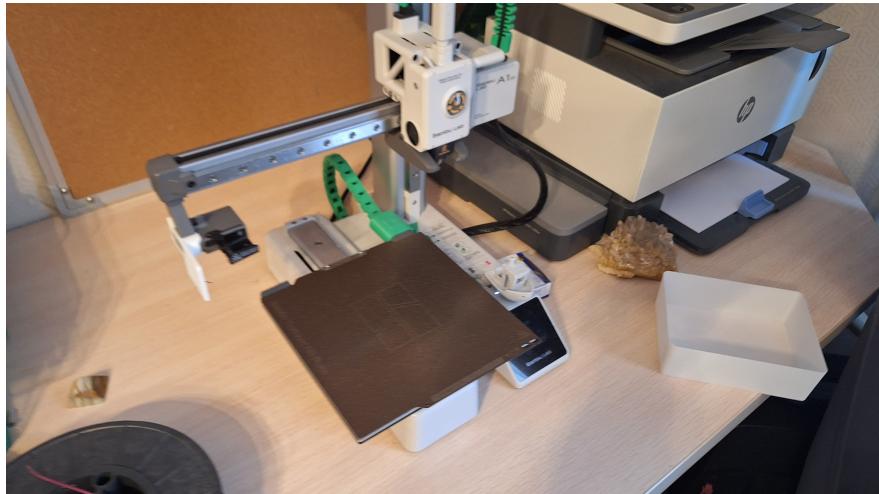


Рис. 2.2: Зображення принтера отримане за допомогою камери телефона

Отримані значення RGB: (89, 65, 55) для столу і (219, 208, 202) для корпусу. Після переходу в HSV отримано (0.05, 0.382, 0.349) і (0.0583, 0.078, 0.859) відповідно. Покладено допустимі відхилення тону кольору і насиченості в 0.1 для столу. Покладено що насиченість для корпусу має бути менша за 0.2, при будь-якому значенні тону. Дані правила застосовані для даного зображення.

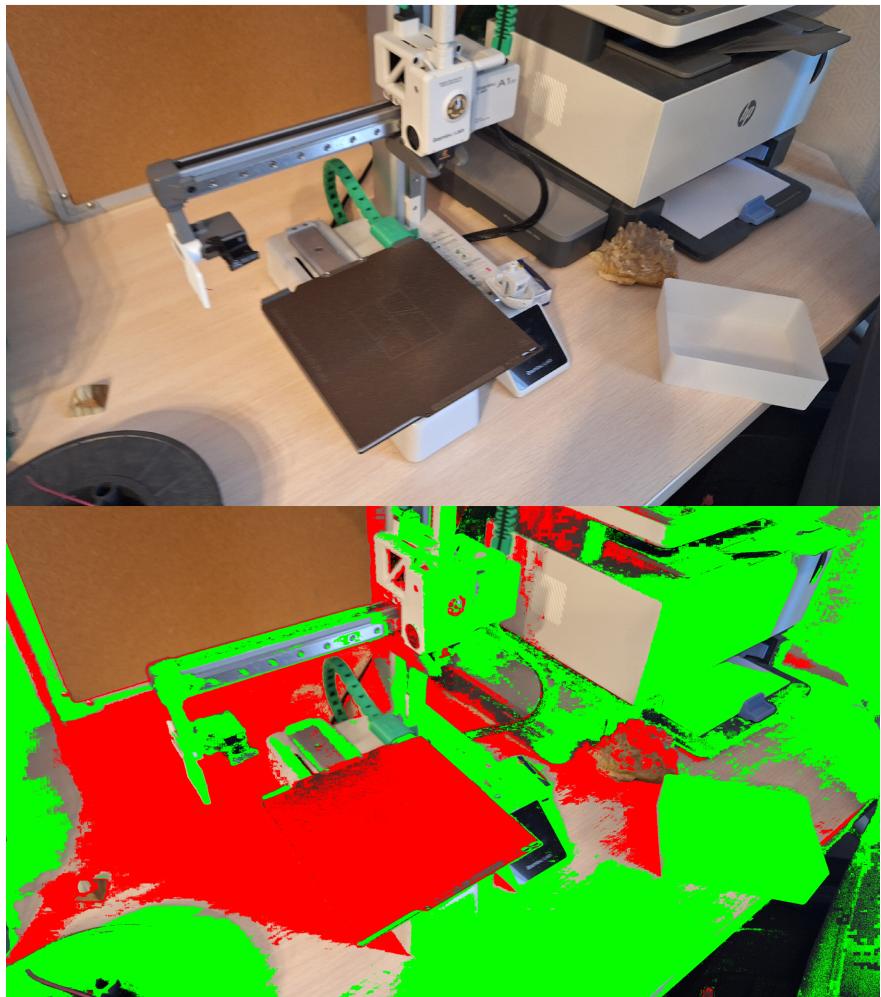


Рис. 2.3: Перший набір правил застосований до зображення: правила для столу червоним, правила для корпусу білим

Застосування даних правил значно зменшило область пошуку столу, проте вона все ще залишалася великою. З порівняння зображень видно, що правила для столу спрацьовують для світлих ділянок, а правила для корпусу – для темних. Щоб уникнути таких спрацювань, було додано правило на яскравість: для столу яскравість має бути меншою за середню на зображенні, а для корпусу – більшою.

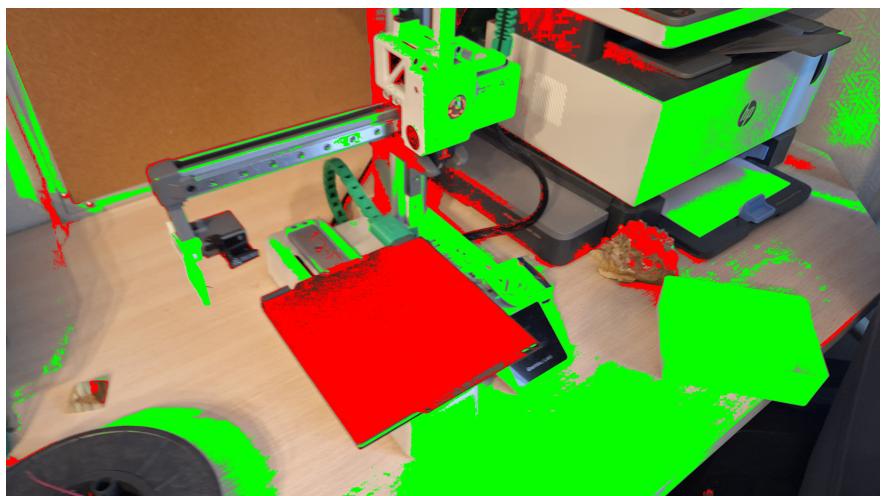


Рис. 2.4: Другий набір правил застосований до зображення

Дане розширення правил ще раз значно зменшило область пошуку. Проте, частина столу не

фарбується в червоний, відповідно необхідно збільшити допустимі відхилення тону і насиченості. Допустимі відхилення збільшено вдвічі.

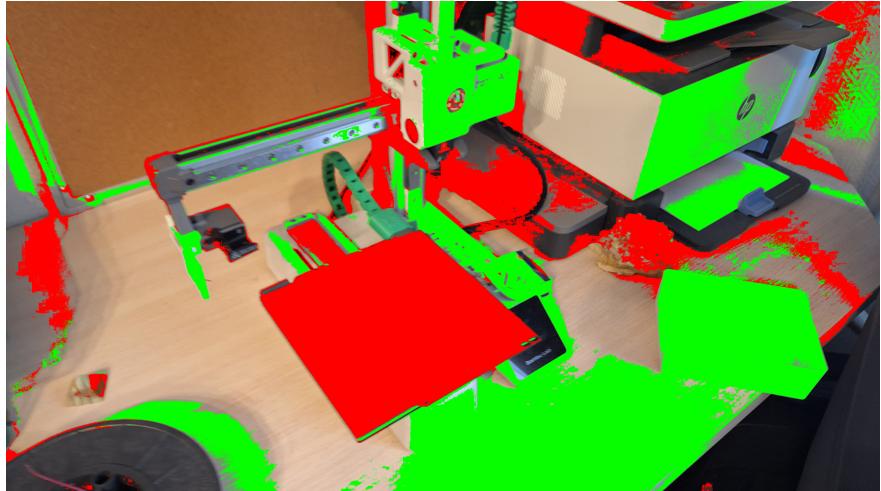


Рис. 2.5: Третій набір правил застосований до зображення

#### 2.4.2. Отримання границь, як джерел контексту

Для отримання контексту у вигляді границь об'єктів необхідно застосувати матричний фільтр границь:

$$kernel = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Даний фільтр застосовано до різних зображень: оригінального, оригінального в HSV та, чорно-білого. Після чого було взято середнє всіх трьох каналів. Для кращого відображення границь, розмір зображення зменшено в 8 разів.

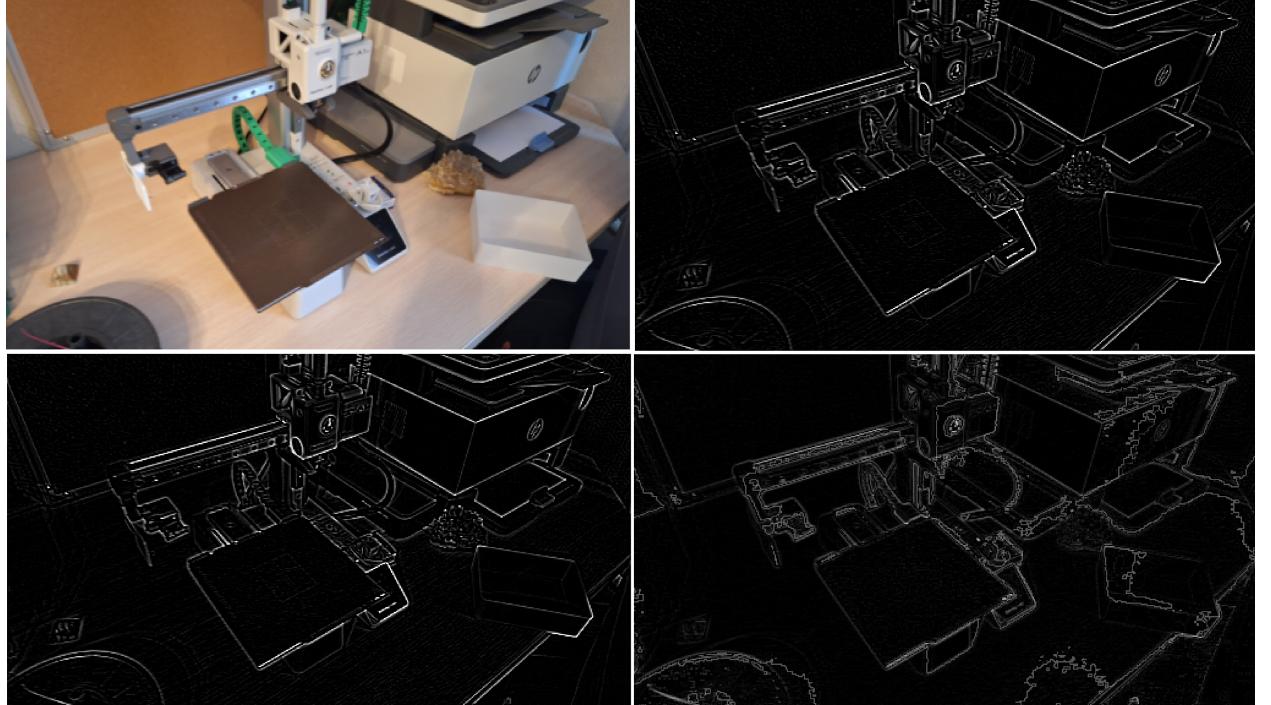


Рис. 2.6: Перший ряд: оригінальне зображення, застосування фільтру до RGB, застосування фільтру до середнього значення каналів RGB, застосування фільтру до HSV

З зображень видно, що фільтр HSV має занадто нечіткі лінії щоб використовуватись далі.

Для утворення чітких границь, необхідно їх перетворити у бінарні значення 0 і 1. Один з більш простих способів такого перетворення – прогове перетворення:

$$1, \text{pixel} \geq \text{threshold}$$

$$0, \text{pixel} < \text{threshold}$$

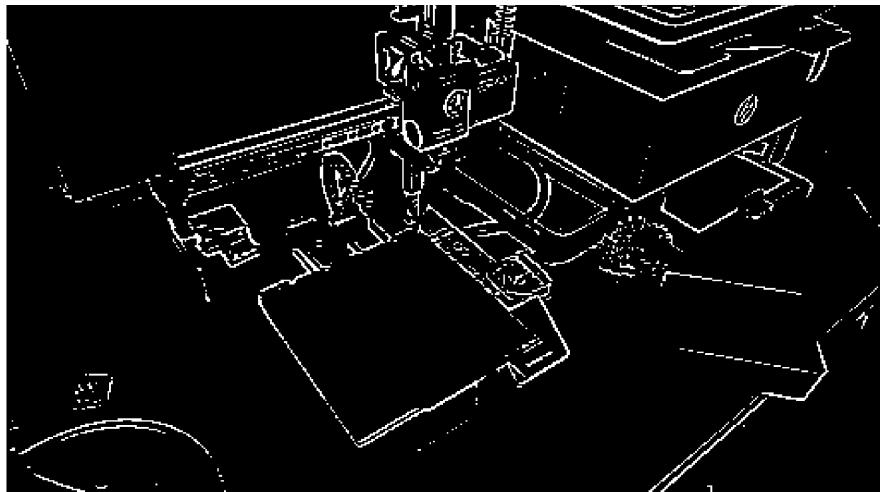


Рис. 2.7: Порогова бінаризація RGB границь з порогом 0.2

#### 2.4.3. Використання попередніх результатів розпізнання

Нехай програма вже успішно змогла розміznати стіл в попередньому моменті часу (наприклад на попередньому кадрі відео), тоді відомо масив точок що входили в регіон столу. Оскільки хід столу вперед-назад не перевищує розміру самого столу, то якась частина попереднього масиву точок мають опинитись в новому регіоні столу. Можна впровадити простий критерій, що хоча б одна точка з попереднього регіону столу, має бути в новому регіоні столу.

Якщо попередні моменти часу відсутні, можна попросити людину, або більш потужний розпізнавач розпізнати початкове положення столу. У разі використання людини користувача/оператора, їй достатньо вказати якусь точку столу, щоб звичайний розпізнавач за цією точкою знайшов стіл на першому зображення.

## 2.5 Створення фітнес-функції для оцінки чотирикутників

#### 2.5.1. Загальні вимоги до фітнес-функції

Фітнес-функція має приймати на вхід вершини чотирикутника, окрім вершин функція може приймати значення вершин в попередній момент часу, початкову точку та константи пов'язані з оброблюваним зображенням.

#### 2.5.2. Функція оцінки квадратності чотирикутника

Квадрат є паралелограмом, з більшості кутів огляду (викривленням лінзи і перспективою для демонстраційних цілей роботи методів можна знахтувати). Паралелограми мають наступні властивості: їх протилежні сторони рівні, їх протилежні сторони паралельні.

Оцінка рівності протилежних сторін:

$$\theta_E = \frac{(E_1 - E_3)^2}{E_1 + E_3} + \frac{(E_2 - E_4)^2}{E_2 + E_4}, \text{ де } E_i \text{ – довжина ребра.}$$

В оцінці рівності протилежних сторін присутнє ділення на суму цих протилежних сторін, інакше оцінка буде працювати по-різному для різних довжин ребер.

Оцінка паралельності протилежних сторін:

$$\theta_\alpha = (\alpha_1 - \alpha_3)^2 + (\alpha_2 - \alpha_4)^2, \text{ де } \alpha_i - \text{кут нахилу ребра в радіанах}, \frac{\pi}{2} \leq \alpha_i \leq \frac{3\pi}{2}.$$

Об'єднання оцінок рівності і паралельності протилежних сторін можна виконати безліччю методів. Простішими з них є сума і множення. Сума оцінок зберігає незалежність оцінок і дозволяє покращувати одну оцінку, навіть якщо інша занулена. Але сума потребує балансування оцінок, інакше алгоритми і моделі будуть намагатись оптимізувати оцінку з найбільшим абсолютним значенням. Множення не має необхідності балансування оцінок, проте має проблеми, якщо якась з оцінок занулюється. Щоб уникнути занулення, до оцінок можна додати невелику константу.

Оцінки об'єднані наступним чином:

$$\theta_{\text{пар.}} = (\theta_E + c_1) \cdot (\theta_\alpha + c_2), \text{ де } c_i - \text{константи уникнення занулення, які необхідно визначити експериментально.}$$

### 2.5.3. Функція оцінки наявності границь поруч з сторонами чотирикутника

Оскільки по периметру столу 3D принтера майже завжди знаходяться границі, можна рахувати кількість пікселів границі поруч з сторонами чотирикутника. Нехай "поруч" це "сторона проходить через піксель". Для визначення пікселів через які проходить відрізок існує алгоритм Брезенхейма [?]. Загальне рівняння прямої через дві точки  $(x_0, y_0)$ ,  $(x_1, y_1)$ :

$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0}$$

Алгоритм Брезенхейма, для всіх 8 напрямків прямої:

1.  $p$  – пустий масив точок;
2.  $\Delta x = |x_1 - x_0|$ ;  $\Delta y = |y_1 - y_0|$ ;
3.  $sx = x_1 \geq x_0$ ;  $sy = y_1 \geq y_0$ ;
4.  $x = x_0$ ;  $y = y_0$ ;
5.  $D = argmax(\Delta x, \Delta y)$  – домінантна вісь за якою відбуваються кроки,  $S$  – інша вісь ( $X, Y = D, S$  або  $Y, X = D, S$ );
6.  $Error = floor(\Delta d/2)$ ;
7. Доки  $d \neq d_1$ :
8. Записати  $(x, y)$  в  $p$
9.  $Error = Error - \Delta s$
10. Якщо  $Error < 0$ , то:  $s = s + ss$ ;  $Error = Error + \Delta s$
11.  $d = d + sd$ ;
12. Повернутись до "Доки";
13. Записати  $(x_1, y_1)$  в  $p$ ;
14. Повернути  $p$ .

Для обрахунку кількості білих пікселів, алгоритм модифіковано наступним чином:

1.  $p = 0$ ;
2.  $\Delta x = |x_1 - x_0|$ ;  $\Delta y = |y_1 - y_0|$ ;
3.  $sx = x_1 \geq x_0$ ;  $sy = y_1 \geq y_0$ ;
4.  $x = x_0$ ;  $y = y_0$ ;
5.  $D = argmax(\Delta x, \Delta y)$  – домінантна вісь за якою відбуваються кроки,  $S$  – інша вісь ( $X, Y = D, S$ , або  $Y, X = D, S$ );

6.  $Error = \text{floor}(\Delta d/2);$
7. Доки  $d \neq d_1$ :
8.  $p = p + \text{Image}[x][y];$
9.  $Error = Error - \Delta s$
10. Якщо  $Error < 0$ , то:  $s = s + ss$ ;  $Error = Error + \Delta s$
11.  $d = d + sd;$
12. Повернутися до "Доки";
13.  $p = p + \text{Image}[x_1][y_1];$
14. Повернути  $p$ .

Маючи алгоритм обрахунку кількості білих точок на відрізку, можна застосувати його для чотирьох сторін чотирикутника, сума результатів може бути функцією оцінки. Але, в такій реалізації оцінка штрафує за збільшення кількості білих точок. Нехай  $B_4$  – сума результатів роботи модифікованого алгоритму Брезенхейма, тоді:

$$\theta = \frac{1}{B_4}$$

оцінка яка винагороджує за кількість білих точок. Проте, така оцінка буде давати однакову винагороду за 1 білий піксель для чотирикутників з периметром 4 і з периметром 100. Щоб цього уникнути, можна домножити на периметр:

$$\theta_{B_4} = \frac{\sum_{i=1}^4 E_i}{B_4}$$

#### 2.5.4. Функція оцінки внутрішніх границь

Оскільки на зображенні цілком ймовірно може бути присутній інший паралелограм, то він може задовольнити попередні оцінки. Якщо цей паралелограм містить паралелограм столу, то навіть використання точок попереднього регіону столу може не допомогти. Якщо один паралелограм містить інший – значить в ньому є білі пікселі границь внутрішнього паралелограму.

Можна побудувати оцінку, яка враховує кількість білих пікселів в чотирикутнику. Щоб побудувати таку оцінку необхідно побудувати алгоритм перевірки належності пікселя до чотирикутника.

Якщо точка знаходиться в чотирикутнику, промінь з цієї точки в будь-якому напрямку перетне чотирикутник. Якщо чотирикутник не опуклий і цей промінь перетне іншу сторону – промінь має перетнати третю сторону щоб залишити чотирикутник. Таким чином, якщо промінь з точки в будь-якому напрямку перетинає сторони чотирикутника один чи три рази – точка знаходиться в чотирикутнику.

Якщо хоча б один промінь з точки не перетинає жодну зі сторін чотирикутника – точка лежить поза чотирикутником. Якщо ж перетинів парна ненульова кількість – промінь спочатку заходить, а потім виходить з чотирикутника.

Таким чином: якщо для довільного променя кількість перетинів парна – точка лежить поза чотирикутником, якщо непарна – в середині. Для алгоритму використано промінь вздовж осі X:

1.  $inside = False;$
2. Для кожної сторони:
3.  $x_i, y_i, x_j, y_j$  – координати вершин сторони;
4. Якщо промінь вздовж X між  $y_i$  та  $y_j$  і якщо точка перетину з прямою сторони в середині сторони – є перетин,  $inside = not inside$ ;
5. Після проходу по всім сторонам – повернути  $inside$ .

Маючи алгоритм належності пікселя до чотирикутника, можна побудувати алгоритм підрахунку білих точок в середині чотирикутника:

1.  $c = 0$ ;
2. Для кожної білої точки на зображенні:
3. Якщо точка в чотирикутнику –  $c = c + 1$ ;

Таким чином, оцінка внутрішніх границь має вигляд  $\theta = c$ . Але така оцінка не враховує розмір чотирикутника, для врахування розміру слід ділити на площину чотирикутника:

$$\theta_c = \frac{c}{S}, \text{ де } d_i - \text{довжина діагоналі, } S - \text{площа чотирикутника}$$

#### 2.5.5. Комбінація функцій оцінки у фінтес-функцію

Одними з найбільш простих способів об'єднань оцінок є додавання і множення. Для того щоб уникнути балансування оцінок між собою, обрано множення:

$fitness = \theta_{\text{пар.}} \cdot (\theta_{B_4} + c_3) \cdot (\theta_c + c_4)$ , де  $c_i$  – константи уникнення занулення, які треба визначити експериментально

## 2.6 Створення первинного набору даних

Для набору даних на камеру телефона знято 50 фотографій принтера в однаковому довкіллі і при однаковому освітленні (різні умови довкілля не розглядаються для демонстрації роботи методів і зменшення часу на навчання моделей). Приклад однієї з таких фотографій на рисунку 2.2.

Розглянувши декілька програм для встановлення розмітки на зображеннях, обрано Label Studio [?] як безкоштовний застосунок. Для зображень створено анотації з чотирикутниками.

Для збільшення датасету, створено копії зображень: трохи повернуті та змінені в розмірах.

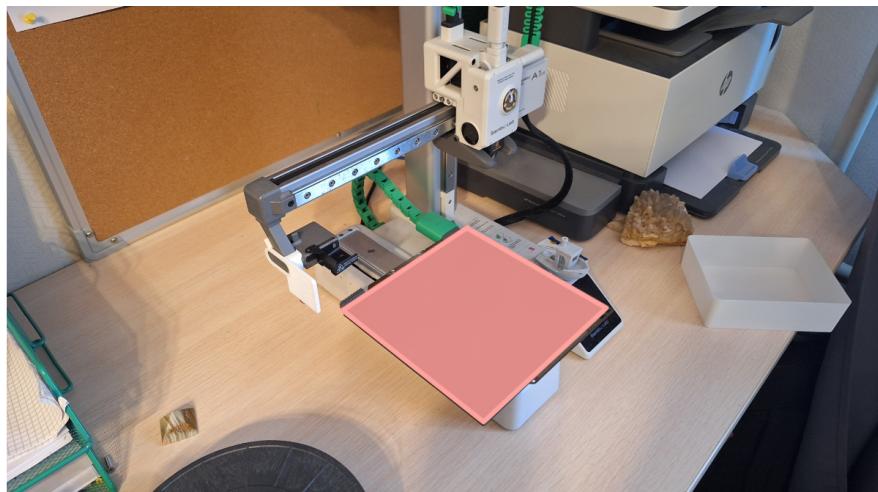


Рис. 2.8: Приклад розміченого зображення

# 3. Розробка програми

## 3.1 Реалізація загальних класів та функцій

### 3.1.1. Робота з зображеннями

Створено клас `ImageContainer`, який містить масив значень пікселів від 0 до 1 у RGB або HSV та має метод для показу зображення на екрані. Для роботи з класом створені функції: перетворення з RGB в HSV, усереднення трьох каналів, застосування матричний фільтр, порогової бінаризації, зміни розміру та перетворення зображення на масив позицій білих пікселів.

Окрім порогової бінаризації, створено функцію яка використовує бінарний пошук для знаходження значення порогу таке, що частка білих пікселів на зображені приблизно дорівнює вказаному аргументу.

### 3.1.2. Реалізація оцінок і фітнес-функції

Створено клас `Quad` який при ініціалізації приймає масив з 8 значень:  $x_1, y_1, x_2, \dots, y_4$ . З цих значень обраховуються: масиви координат вершин (вершини сортуються за порядком обходу, щоб уникнути чотирикутників які виглядають як мінімалістичний піщаний годинник), довжин і кутів нахилу сторін; площа і периметр чотирикутника.

Оголошуються функції оцінок і конструктор фітнес функції, який бере на вхід константи і повертає функцію яка на вхід приймає лише масив з восьми змінних значень.

## 3.2 Алгоритми знаходження чотирикутника з найкращим значенням фітнес-функції

### 3.2.1. Повний перебір

Алгоритм повного перебору дозволяє достовірно обчислити найкращий чотирикутник, але за рахунок використання великої кількості обчислювального часу. Щоб такий алгоритм міг закінчити роботу щонайменше в поточному сторіччі, нобхідно зменшувати розмір вхідного зображення і використовувати паралелізацію.

Створено функцію, яка для відповідного масиву білих пікселів робить повний перебір всіх строго зростаючих за індексом комбінацій білих пікселів.

Для подальших експериментів  $c_i = 1$ .

Для оригінального зображення (4080x2296 пікселів) для перебору необхідно 3.8 мільярдів років. Для зменшеного в 16 разів зображення – 2312 років. Для зменшеного в 64 рази – 62 дні. Для зменшеного в 256 разів – 90 секунд.

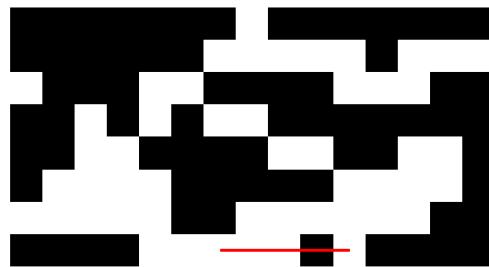


Рис. 3.1: FullCalcSmall роботи повного перебору білих пікселів для границь зменшеного в 256 разів зображення



Рис. 3.2: Результат роботи повного перебору на оригінальному зображення

Цей алгоритм не здатний визначити положення столу за менш ніж декілька місяців, тож його використання недоцільне.

### 3.2.2. Випадковий перебір

Випадковий перебір простий в реалізації, легкий в паралелізації, не потребує великої кількості часу, але й не гарантує результат.

Створено функцію яка робить вказану кількість випадкових перевірок і повертає найкраще значення фітнес-функції, значення координат і графік зміни фітнес-функції впродовж перебору.

Обраховано випадковий перебір протягом  $10^5$  для зменшеного в 32 рази зображення:

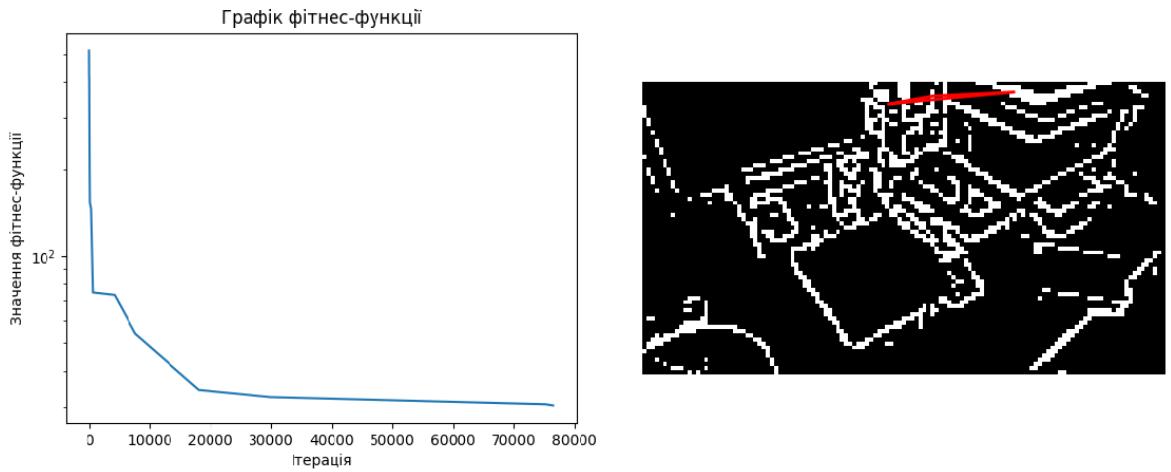


Рис. 3.3: Зліва графік збіжності фітнес-функції до 0, справа знайдений чотирикутник на фоні границь

З графіку видно, що випадковий перебір не в змозі отримати задовільні значення фітнес-функції.

### 3.2.3. PSO та покращення фітнес-функції

Створено функцію яка застосовує ройовий алгоритм PSO для вхідного зображення з заданими набором гіпер параметрів і кількістю ітерацій.

Основні параметри	Значення
Прискорення до найкращого власного значення	1
Прискорення до найкращого значення популяції	1.5
Розмір популяції	250
Максимальна швидкість	100
Збереження швидкості (анти гальмування)	90%

Табл. 3.1: Таблиця значень гіпер параметрів для перших тестових запусків PSO

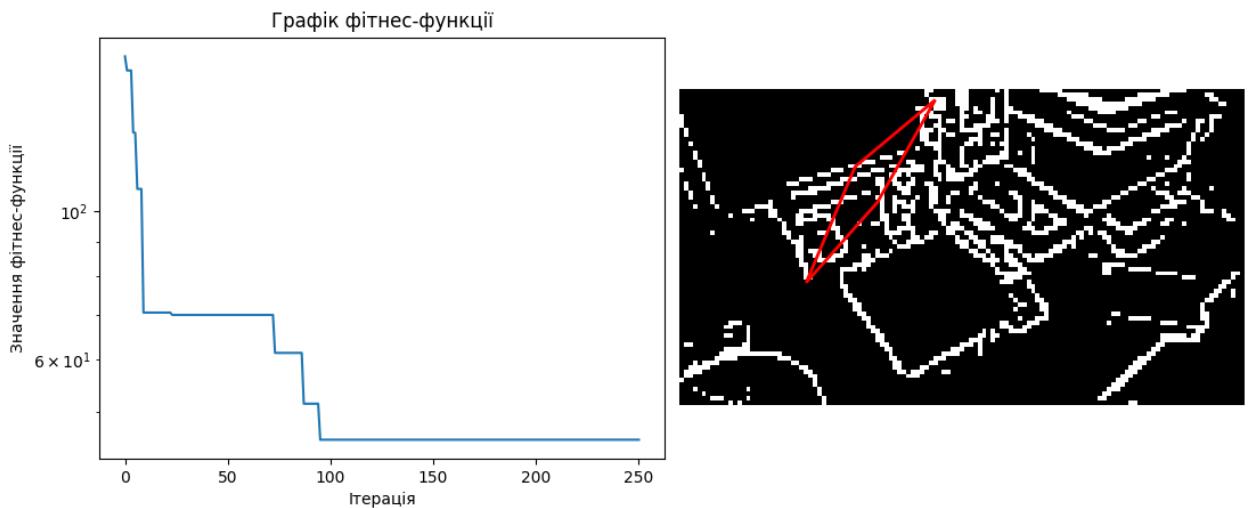


Рис. 3.4: Результат роботи PSO для зменшеного в 32 рази зображення протягом 250 ітерацій: зліва графік збіжності, справа отриманий чотирикутник

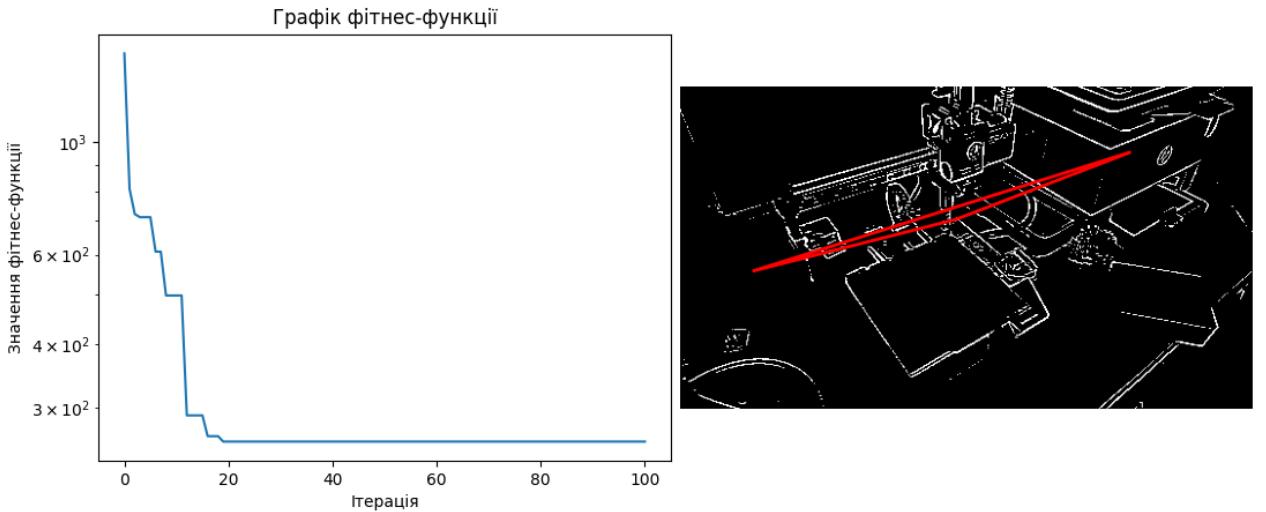


Рис. 3.5: Результат роботи PSO для зменшеного в 8 разів зображення протягом 100 ітерацій: зліва графік збіжності, справа отриманий чотирикутник

З форми чотирикутника видно, що фітнес функція не враховує те, що ромб з малим кутом непогано задовільняє оцінки паралелограма. Додано оцінку, яка штрафує за різницю розміру діагоналей:

$$\theta_{2D} = \frac{\left(D_1 - \frac{D_1+D_2}{2}\right)^2}{D_1} + \frac{\left(D_2 - \frac{D_1+D_2}{2}\right)^2}{D_2}.$$

Також, модифіковано фітнес-функцію:

$$fitness = \theta_{\text{пар.}}^2 \cdot (\theta_{B_4} + c_3)^2 \cdot (\theta_c + c_4)^2 \cdot (\theta_{2D} + c_5)^2$$

Але подальші декілька десятків тестових запусків і незначних змін фітнес-функції не привели до задовільних результатів.

### 3.2.4. Використання контексту для покращення фітнес-функції

Для покращення результатів роботи алгоритмів, використано раніше створені методи отримання контексту. Задля отримання границь використовується маска для правила кольору столу, у фітнес функцію додано індикатор наявності точки столу в чотирикутник.

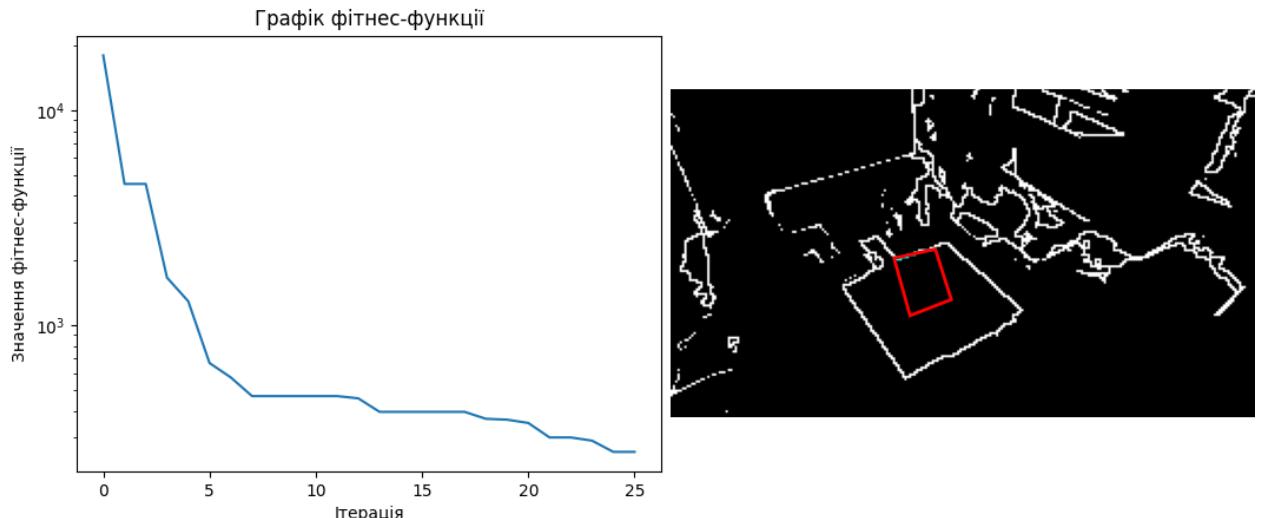


Рис. 3.6: Результат роботи PSO для зменшеного в 16 разів зображення з границями контексту кольору столу протягом 100 ітерацій: зліва графік збіжності, справа отриманий чотирикутник

Оскільки робота з границями не дала бажаних результатів, використано пікселі з контекстом кольору столу і модифіковано  $\theta_c = \frac{1}{c}$  для винагородження за кількість пікселів в середині чотирикутника. Якщо  $\theta_c$  майже 1 – більше винагороджувати за площину. Ігнорувати кількість пікселів на границі.

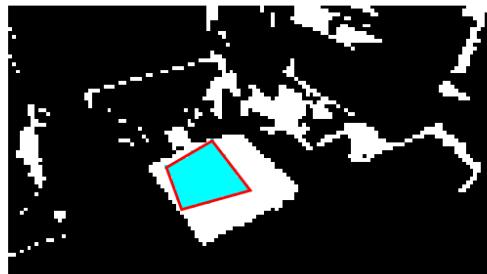


Рис. 3.7: Результат роботи PSO з 1000 частинок для зменшеного в 32 рази зображення з контекстом кольору протягом 50 ітерацій

### 3.2.5. Використання генетичного алгоритму

#### Реалізація генетичного алгоритму

Генетичний алгоритм реалізовано наступним чином: масив з масивів 8 координат кожної ітерації проходить кросовер, мутацію і селекцію. Кросовер випадковим чином обирає двох батьків і для кожної координати створює випадкове значення в межах значень координат батьків. Мутація з заданою ймовірністю і силою може посунути і повернути весь чотирикутник та випадковим чином трохи поруhatи координати,  $\frac{1}{10}$  розміру популяції не підлягає мутації. Селекція сортуює популяцію за значенням фітнес-функції та залишає лише найкращі хромосоми. Задля пришвидшення роботи коду, обчислення фітнес-функції при селекції паралелізовано.

Також, існує опція збільшення початкового набору хромосом, щоб алгоритм починав роботу з більш кращими хромосомами

#### Використання генетичного алгоритму

Виконано декілька обчислень для фітнес-функції для маски контексту столу і контексту точки столу. Ціль обчислень – встановити чи може фітнес-функція що базується виключно на оцінці  $\theta_c$  і контексті точки столу дати задовільний результат.

Обчислення показали, що така фітнес-функція провокує дуже вузькі чотирикутники, які захоплють по 2 пікселі, маючи середню ширину 1. Додавання оцінки рівності діагоналей прибрало цю проблему, але код генерував трикутники. Додавання оцінки рівності протилежних кутів дало наступний результат:

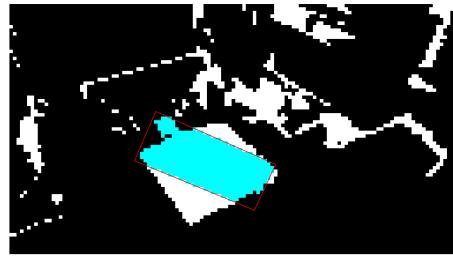


Рис. 3.8: Результат роботи генетичного алгоритму для фітнес-функції без перевірки пікселів на сторонах

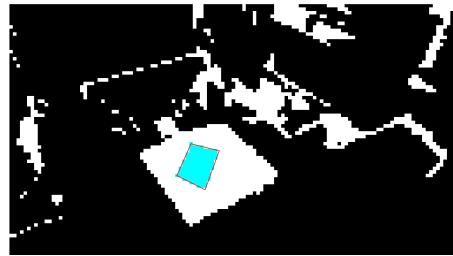


Рис. 3.9: Результат роботи генетичного алгоритму для фітнес-функції з перевіркою пікселів на сторонах

Після ще декількох експериментів з фітнес функцією та оцінками зроблено висновок, що побудовані оцінки та фітнес-функція не здатні забезпечити бажаний результат.

### 3.3 Побудова нейронних мереж для вирішення задачі

#### 3.3.1. Загальна структура нейронних мереж

Вихідним шаром нейронної мережі має бути шар що повертає 8 значень координат. Вхідний шар – довільний для зображень, може приймати на вхід оригінальне зображення, або контекст.

#### 3.3.2. Нейронні мережі що використовують оригінальні зображення

##### **Convolutional neural networks навчені на оригінальних зображеннях**

Використано "класичну" структуру CNN – каскад з трьох пар Conv2D та MaxPooling, один внутрішній Dense шар і один вихідний Dense. Задано параметр BASE, кількість фільтрів для Conv2D:  $2^{BASE}, 2^{BASE+1}, 2^{BASE+2}$ ; кількість нейронів внутрішнього Dense –  $2^{BASE+3}$ .

Оригінальні зображення мають великий розмір, тож доцільно зменшувати їх перед подачею в нейронну мережу. Нехай розмір вхідного шару – 720x480 пікселів.

Натреновано моделі для BASE від 1 до 4.

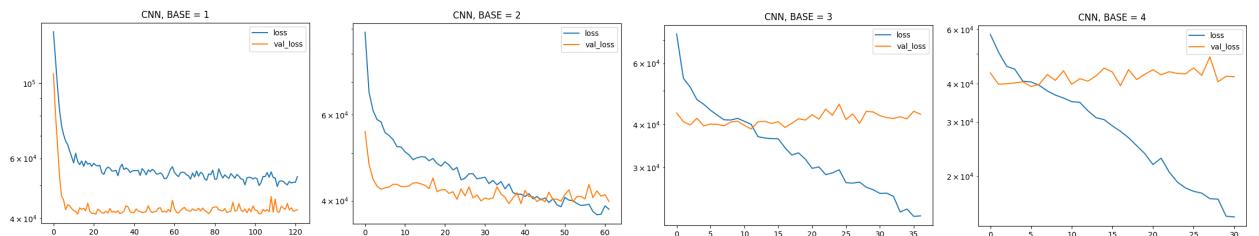


Рис. 3.10: Графіки функцій втрат для чотирьох моделей



Рис. 3.11: Приклад розпізнання столу однією з моделей

Після вхідного шару додано шар, який застосовує контекстне правило столу до зображення і повертає маску.

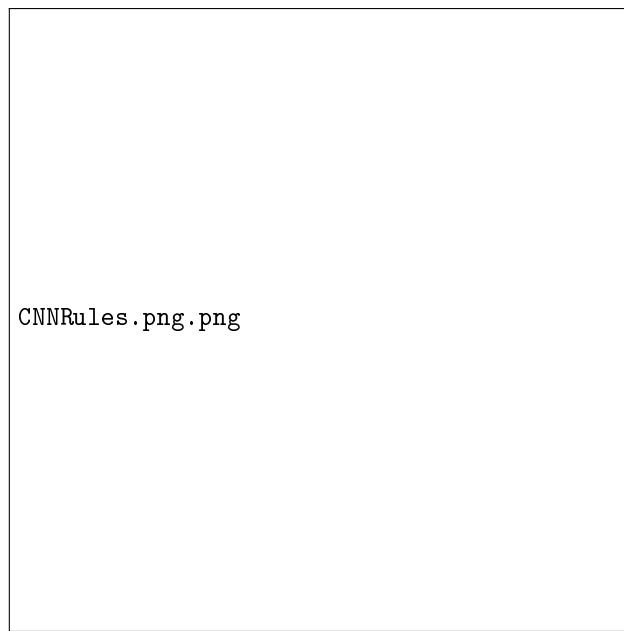


Рис. 3.12: Графік функції втрат для моделі з конекстним правилом

Як видно з графіку, додавання такого шару не допомогло моделі

## Модифікації існуючих мереж

## **4. Висновки**

# Бібліографія

- [1] Bresenham's line algorithm [Електронний ресурс]. – Режим доступу: <https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html> (дата звернення: 13.06.2025).
- [2] Label Studio. Documentation guide [Електронний ресурс]. – Режим доступу: <https://labelstud.io/guide> (дата звернення: 13.06.2025).
- [3] NumPy: Linear algebra (numpy.linalg) [Електронний ресурс]. – Режим доступу: <https://numpy.org/doc/2.2/reference/routines.linalg.html> (дата звернення: 13.06.2025).