**Детектор друкованого тексту**

**Ідея проекту:** на вхід програми подається зображення друкованого тексту. Результатом виконання програми є електронний текст.

**Реалізація проекту:** реалізація проекту містить три основні частини-модулі:

* Визначення та розділення окремих літер та символів
* Класифікація символів
* Апостеріорна обробка результатів

**Загальні відомості:** Для роботи будемо використовувати растрові зображення. Для математичного моделювання можна буде розглядати зображення як функцію залежності інтенсивності від координат

де – вектор пікселя в певному кольоровому просторі.

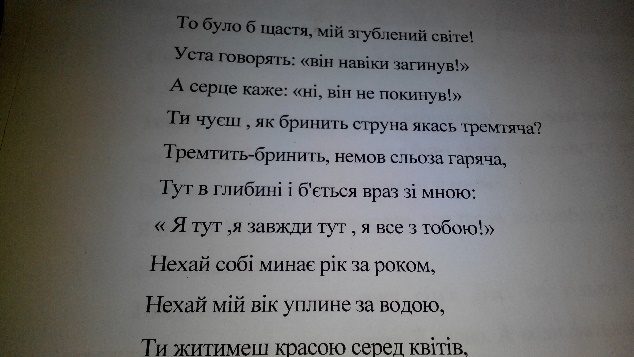
**Визначення та розділення окремих літер та символів**

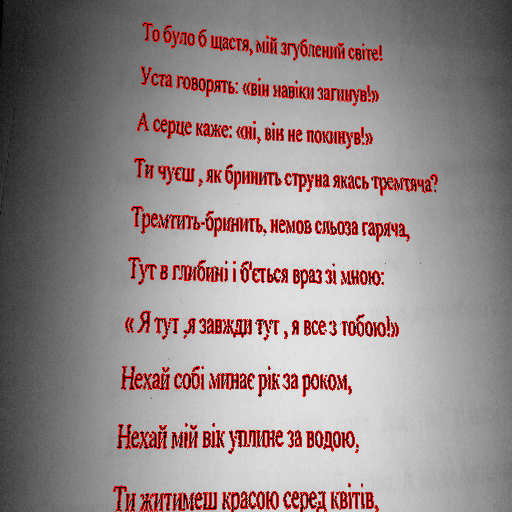
Будемо розглядати загальний випадок чорного шрифту на білому фоні. Проблема полягає в розділенні символів, які не з’єднані між собою суцільною літерою. Для цього можна використати порогові операції ([threshold operations](https://en.wikipedia.org/wiki/Thresholding_(image_processing))).

Інший варіант визначення позицій літер може базуватись на знаходженні кутів літер (edges). Для цього можна використовувати часткові похідні

де і градієнт по горизонталі та вертикалі відповідно. Враховуючи дискретність функції зображення, похідна буде обчислюватись як різниця сусідніх елементів

Шукаючи локальні максимуми та мінімуми після диференціювання зображення ми отримаємо координати тих точок, які є межею зміни інтенсивності. У випадку тексту – це змінна чорного кольору на білий, що є доволі великою зміною градієнту.

Взагалі кажучи, для досягнення кращих результатів визначення кутів на зображення можна використовувати і складніші техніки, які базуються на похідній другого порядку ([оператор Лапласа](https://en.wikipedia.org/wiki/Laplace_operator)), аналізі величини градієнту на порогових операцій ([алгоритм Кенні](https://en.wikipedia.org/wiki/Canny_edge_detector)) тощо. 



**Класифікація символів**

Для класифікації символів варто використати один з класифікаторів. Для вибору слід провести тестування точності визначення для кожного з них на конкретних символах з великого обсягу тестових даних. Дані потрібно розділити в пропорції 4:1 (80% і 20%) для навчання та тестування точності відповідно. Серед кандидатів є такі класифікатори

* K Nearest Neighbor

Досить простий та швидкий класифікатор. Кожен приклад навчання подається у вигляді вектора-точки (зображення розкладається по рядках або стовпцях). Таким чином зображення розміру утворюватиме точку з простору , (ℒ - кількість каналів в певному колірному просторі). Кожна точка має певну мітку, до якого класу вона відноситься.

Тестове зображення відповідної розмірності переводиться у точку і повним перебором обраховується відстань до усіх інших точок. Вибирається K найближчих точок. Аналізуючи класи цих точок, можна прийти до висновку належності нашого зображення до певного класу.

* Support Vector Machine (SVM)

Дозволяє розділити два кластери, шукаючи опорні вектори серед даних для навчання. Проводячи пряму (у випадку двовимірного простору) цей алгоритм розділяє дані на два класи. У випадку нелінійної роздільності можна перетворювати простір різними алгоритмами та розділяти нелінійні кластери. Дає погані результати у випадку лінійної нероздільності класів.

* Artificial Neuronal Networks

Швидкий у використанні класифікатор. Потребує великі ресурсні затрати на навчання. Кожен нейрон приймає на вході n-вимірний вектор та повертає дійсне число, яке може бути входом іншого нейрона. Кожен нейрон містить свої внутрішні коефіцієнти, які підбираються в процесі навчання.

**Апостеріорна обробка результатів**

Для покращення результатів виявлення та класифікації літерів можна використовувати словники для перевірки коректності слів. Якщо конкретне слова у словнику не знайдено, можна порахувати ймовірності помилки утворення даного слова. Якщо ця ймовірність достатньо велика, то варто редагувати результат та, можливо, внести зміни у класифікатор.

**Перспективи розвитку**

В майбутньому проект можна буде розвивати у декількох напрямках

* Розпізнавання інших мов
* Розпізнавання символів
* Робота з неякісними фотографіями
* Розпізнавання рукописного тексту
* Real-Time розпізнавання

**Засоби розробки**

Мова програмування: C++/C# (Core/GUI)

Сторонні бібліотеки та розширення: [OpenCV](http://opencv.org/) (C++)

Середовище: Visual Studio 2015