Ofnir

OCR Project Data

Contact information

Tsz Kiu Wong (24373441) - jackywong816@gmail.com

Gens Kwong (24001069) - gensok@gmail.com

Kris Chen (24204478) - Krischen2003@gmail.com

Abstract

Ofnir is your always available translation tool, sitting in your system tray ready to spring into action. Just crop out a portion of your screen, and within seconds, you’ll get an interactive UI with the precise translations as seamlessly overlaid on the original text. Hover to see the meaning, click to swap back!

Whether you’re a studious researcher, an avid reader, or a language learner. Ofnir delivers lightning fast, highly accurate, and a smooth experience leaving sluggish OCR tools in the dust. Say goodbye to a sluggish delays and hello to effortless understanding.

Tentative Schedule estimates

Phase 1: Core UI/Functionalities (~ 1 - 3 days)
Basic UI, Image Cropping capabilities.

Phase 2: OCR Implementation (~ 1 week)
Integrating Tesseract OCR to process cropped images and outputting them for testing.

Phase 3: Image Preprocessing & Error Handling (~ 1 - 1.5 weeks)
Implementing OpenCV filters and error handling for low confidence levels from Tesseract or no text.

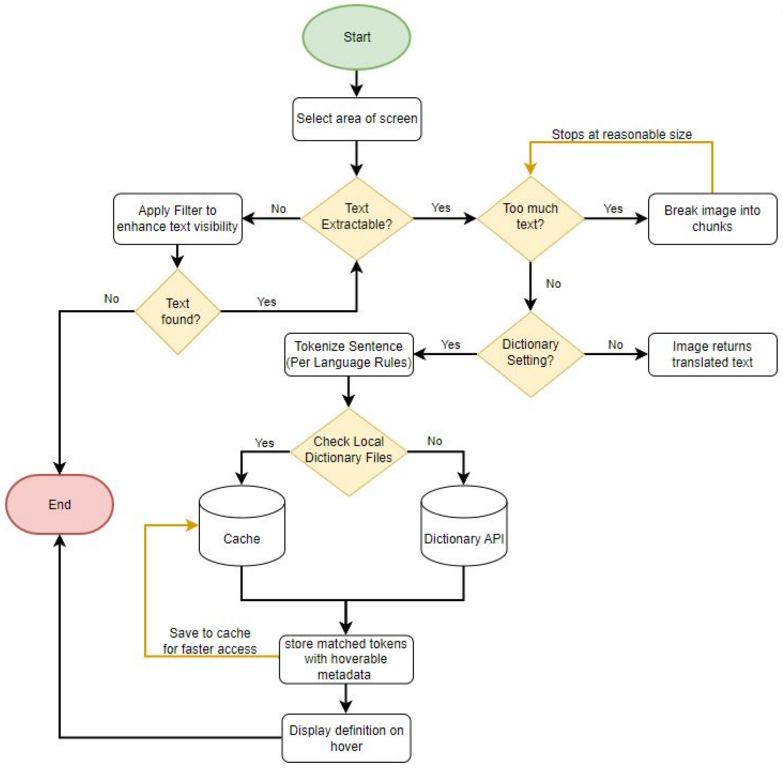
Phase 4: Tokenization system (~ 1.5 - 2 weeks)
Implement a system to tokenize sentences according to set languages.

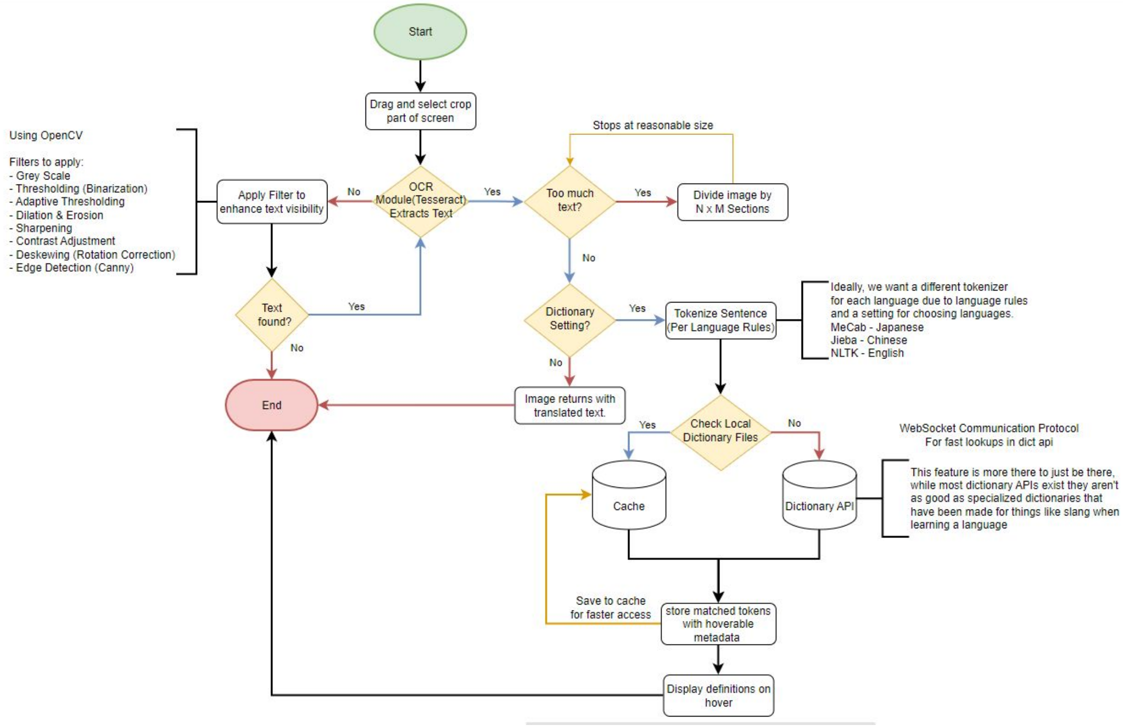
Phase 5: Dictionary API Integration & UI Adjustments (~ 1.5 - 2 weeks)
Integrate an API for word lookup, store it in corresponding tokens to show up on hover and improving UI.

Phase 6: Caching system & Performance Optimization (~ 1.5 - 2 weeks)
Implement Caching of results to for more performance and general performance optimizations.

Phase 7: Final Optimization, Testing & Refinements (~ 1 - 2 weeks)
Polishing, testing and debugging. Final UI adjustments while adding necessary settings.

Diagrams


Start
Select area of screen
Text Extractable?
Yes
No
Apply Filter to enhance text visibility
Text found?
Yes
No
Too much text?
Yes
Break image into chunks
Stops at reasonable size
No
Dictionary Setting?
Yes
No
Image returns translated text
Tokenize Sentence (Per Language Rules)
Check Local Dictionary Files
Yes
No
Cache
Dictionary API
store matched tokens with hoverable metadata
Save to cache for faster access
Display definition on hover
End


Start
Drag and select crop part of screen
OCR Module(Tesseract) Extracts Text
No
Yes
Apply Filter to enhance text visibility
Text found?
Yes
No
Too much text?
Yes
Divide image by N x M Sections
Stops at reasonable size
No
Dictionary Setting?
Yes
No
Tokenize Sentence (Per Language Rules)
Image returns with translated text.
End
Check Local Dictionary Files
Yes
No
Cache
Dictionary API
store matched tokens with hoverable metadata
Save to cache for faster access
Display definitions on hover

Using OpenCV

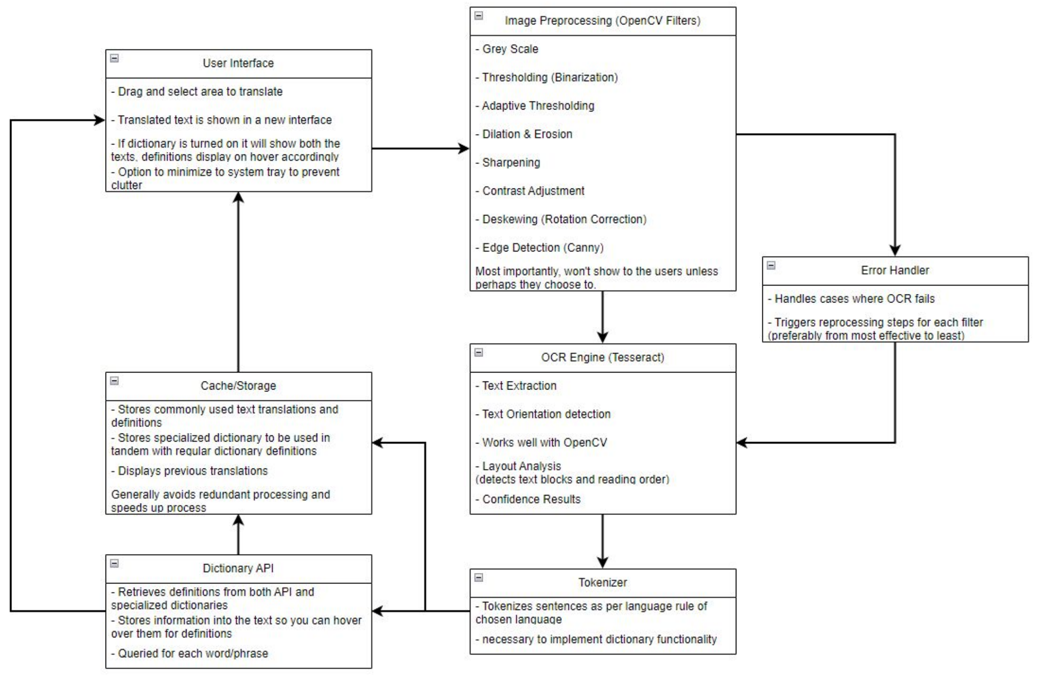
Filters to apply:
- Grey Scale
- Thresholding (Binarization)
- Adaptive Thresholding
- Dilation & Erosion
- Sharpening
- Contrast Adjustment
- Deskewing (Rotation Correction)
- Edge Detection (Canny)

Ideally, we want a different tokenizer for each language due to language rules and a setting for choosing languages.
MeCab - Japanese
Jieba - Chinese
NLTK - English

WebSocket Communication Protocol
For fast lookups in dict api

This feature is more there to just be there, while most dictionary APIs exist they aren't as good as specialized dictionaries that have been made for things like slang when learning a language

Diagrams and extra info



User Interface

- Drag and select area to translate
- Translated text is shown in a new interface
- If dictionary is turned on it will show both the texts, definitions display on hover accordingly
- Option to minimize to system tray to prevent clutter

Image Preprocessing (OpenCV Filters)

- Grey Scale
- Thresholding (Binarization)
- Adaptive Thresholding
- Dilation & Erosion
- Sharpening
- Contrast Adjustment
- Deskewing (Rotation Correction)
- Edge Detection (Canny)

Most importantly, won't show to the users unless perhaps they choose to.

Error Handler

- Handles cases where OCR fails
- Triggers reprocessing steps for each filter (preferably from most effective to least)

OCR Engine (Tesseract)

- Text Extraction
- Text Orientation detection
- Works well with OpenCV
- Layout Analysis (detects text blocks and reading order)
- Confidence Results

Tokenizer

- Tokenizes sentences as per language rule of chosen language
- necessary to implement dictionary functionality

Cache/Storage

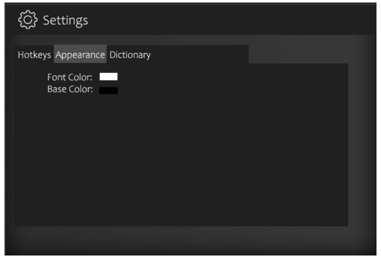
- Stores commonly used text translations and definitions
- Stores specialized dictionary to be used in tandem with regular dictionary definitions
- Displays previous translations

Generally avoids redundant processing and speeds up process

Dictionary API

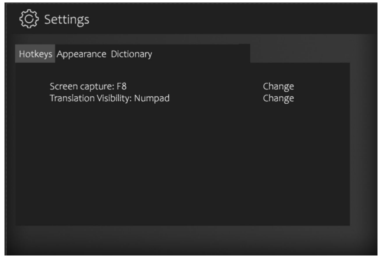
- Retrieves definitions from both API and specialized dictionaries
- Stores information into the text so you can hover over them for definitions
- Queried for each word/phrase

Concept Visualization


Settings

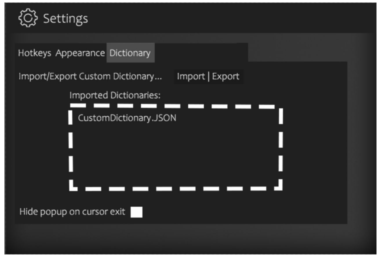
Hotkeys Appearance Dictionary

Font Color: ⬜
Base Color: ⬛


Settings

Hotkeys Appearance Dictionary

Screen capture: F8 Change
Translation Visibility: Numpad Change


Settings

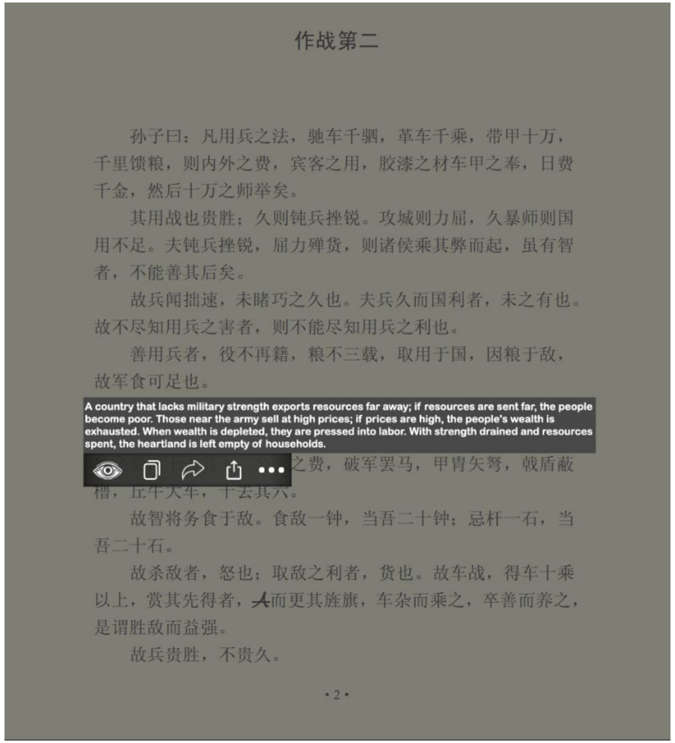
Hotkeys Appearance Dictionary

Import/Export Custom Dictionary... Import | Export

Imported Dictionaries:

CustomDictionary.JSON

Hide popup on cursor exit ⬜


作战第二

孙子曰：凡用兵之法，驰车千驷，革车千乘，带甲十万，千里馈粮，则内外之费，宾客之用，胶漆之材车甲之奉，日费千金，然后十万之师举矣。

其用战也贵胜：久则钝兵挫锐。攻城则力屈，久暴师则国用不足。夫钝兵挫锐，屈力殚货，则诸侯乘其弊而起，虽有智者，不能善其后矣。

故兵闻拙速，未睹巧之久也。夫兵久而国利者，未之有也。故不尽知用兵之害者，则不能尽知用兵之利也。

善用兵者，役不再籍，粮不三载，取用于国，因粮于敌，故军食可足也。

A country that lacks military strength exports resources far away; if resources are sent far, the people become poor. Those near the army sell at high prices; if prices are high, the people's wealth is exhausted. When wealth is depleted, they are pressed into labor. With strength drained and resources spent, the heartland is left empty of households.

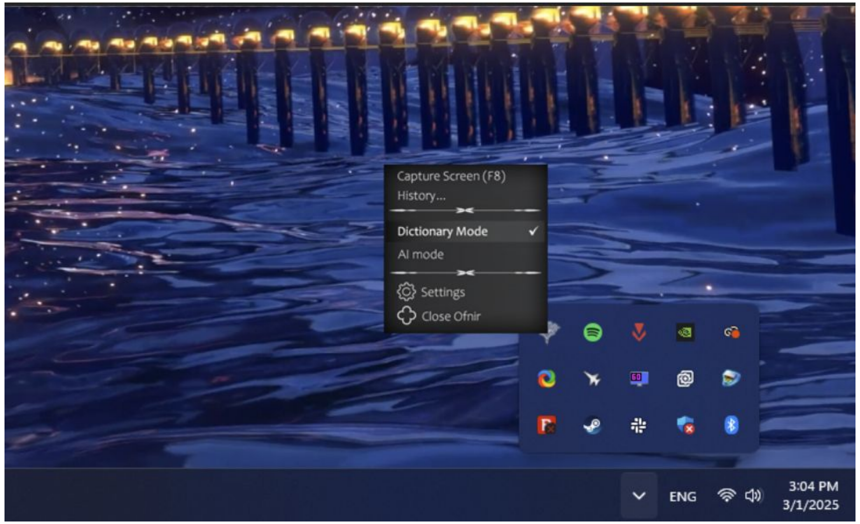
之费，破军罢马，甲胄矢弩，戟盾蔽橹，丘牛大车，十去其六。

故智将务食于敌。食敌一钟，当吾二十钟；忌杆一石，当吾二十石。

故杀敌者，怒也；取敌之利者，货也。故车战，得车十乘以上，赏其先得者，而更其旌旗，车杂而乘之，卒善而养之，是谓胜敌而益强。

故兵贵胜，不贵久。

• 2 •



Tools

Software:

- API services; Dictionary API’s and AI API’s for translation purposes, as well as Tesseract & OpenCV
- Electron.js (Frontend framework)
- Node.js (Backend framework)
- Primary Language: Java

Misc:

- Github project (for project management)
- Github (for repository and skeleton / version control)
- IDE

Datasets

- API Based Translation data: Our translation data will come from the use of dictionary API's that provides us the translation users request.
- Input data: The data our app will receive through the means of scanned documents, PDFs, images containing characters, hovered texts in digital documents, etc.
- Log dataset: Users can have a record of the user's past translations by storing it locally that provides a UI display of the source language, translated language, the time of translation, etc.
- Misc: A potential OCR training dataset that helps our app recognize the more difficult/complicate characters such as handwritten characters.

Use case Scenarios

1. Michael, a software engineer, is using a foreign-language UI tool for work. The interface is entirely in Chinese, and he doesn’t understand some of the settings and buttons. Instead of searching for a manual, he takes a screenshot of the software UI and within seconds, he receives a translated version of the UI labels, helping him navigate the software more effectively / efficiently.
2. Emma is an avid reader of Japanese light novels and Korean webtoons, but many of them haven’t been officially translated. She finds an interesting Korean web novel on a website, but she struggles to understand the text. Instead of relying on browser-based translators (which don’t work on images), she screenshots the text using the app and it gives her the translation.
3. Jason is playing a Japanese Visual Novel Game, but the game has no official English localization. He barely understand the dialogue, Instead of taking a screenshot and extracting text from a website then going to another to translate, he uses the app and instantly gets the translated text without having to leave the game.

Project repo & management board

https://github.com/KrisC2003/OCRProject

https://github.com/users/KrisC2003/projects/2