



RAPPORT DE VEILLE TECHNOLOGIQUE

TrackMyPrompts





2024-2025

HUET CHAPUIS Martin
HAMON Louis
REBOURS Maxence
NATIVELE Elliot

Sommaire

1. Conception du logiciel	4
2. Interface utilisateur	6
3. Interprétation des prompts (détection d'objets)	8
4. Reconnaissance vocale (prompts multimodaux)	10
5. Base de données (Encyclopédie + Historique)	12
6. Transformation de l'application en fichier exécutable	14

Légende des couleurs :

	Très bien
	Bien
	Moyen
	Mauvais, possiblement éliminatoire

Rappel du contexte

Le projet vise à développer une application de détection et de suivi d'objets à partir d'images et de vidéos, destinée à des utilisateurs non-experts. L'objectif est de créer une interface très intuitive, permettant aux utilisateurs d'interagir facilement avec l'IA, de charger des images et vidéos, de saisir des prompts (textuels et vocaux), et d'obtenir des résultats sans écrire de code.

Les besoins du projet incluent :

1. **Interface utilisateur** : Intuitive et simple d'utilisation, permettant l'importation de fichiers ainsi que l'ajustement des hyperparamètres et la gestion de plusieurs modèles de détection à l'aide d'un mode expert.
2. **Traitement d'images et de vidéos** : Capable de traiter des images et vidéos en temps réel pour une détection rapide et précise.
3. **Interprétation des prompts** : Intégrer un système permettant la formulation de prompts par la voix, convertis en texte pour l'analyse.
4. **Base de données** : Gestion d'une encyclopédie contenant la liste des classes que l'IA peut détecter ainsi d'un historique des résultats.
5. **Installation de l'application** : Les utilisateurs pourront installer l'application sous la forme d'un fichier exécutable.

Voici les différents outils que nous pouvons utiliser pour chaque besoin :

- **Conception du logiciel :**
 - QT Designer
 - QT Design Studio
- **Interface utilisateur :**
 - QT (écrit en Python) + Qss
 - Python backend + QML
- **Interprétation de prompts et traitement des données :**
 - YOLO World
 - Faster R-CNN
 - SSD
- **Reconnaissance de voix :**
 - Whisper
 - CMU Sphinx
 - Google Speech Recognition
 - Snowboy Hotword Detection
- **Base de donnée**
 - MySQL
 - SQLite
- **Création d'un fichier exécutable**
 - Cx Freeze
 - PyInstaller

1. Conception du logiciel

Introduction :

Le besoin principal est de choisir un outil de conception d'interface qui permette de créer une application graphique moderne et performante pour des utilisateurs non-experts.

Présentation des solutions :

- **Qt Designer** : C'est un outil visuel permettant de créer des interfaces graphiques simplement. Il est facile à prendre en main, mais reste limité en termes de conception graphique avancée. Il convient pour des designs simples, mais il offre peu d'options pour intégrer des animations complexes ou des transitions. Qt Designer est bien intégré avec PyQt et est totalement open-source.
- **Qt Design Studio** : Cet outil va plus loin en termes de fonctionnalités graphiques. Il permet non seulement de concevoir des interfaces complexes, mais aussi d'ajouter des animations et des transitions. Qt Design Studio est bien adapté pour des projets plus sophistiqués nécessitant des interactions avancées. Il est compatible avec QML et PyQt, mais certaines fonctionnalités avancées nécessitent l'achat de plugins.



Critères de comparaison :

- Facilité d'utilisation
- Complexité des designs possibles
- Compatibilité avec Python
- Licence



Comparaison des solutions :

Critères	Qt Designer	Qt Design Studio
Facilité d'utilisation		
Complexité des designs		
Compatibilité avec Python	(Une fois traduit en python, pas de retour en arrière)	
Licence		

Conclusion :

Bien que **Qt Designer** soit plus simple, **Qt Design Studio** permet d'avoir plus de possibilité et permet de travailler avec des fichiers QML. Nous allons donc utiliser **Qt Design Studio**



2. Interface utilisateur

Introduction :

L'interface doit permettre aux utilisateurs d'importer des images/vidéos, de saisir des prompts, et de visualiser les résultats de la détection d'objets. Il est important que l'interface soit réactive et modernisée pour des utilisateurs non-experts.

Description des solutions :

- **Qt + QSS** : Cette solution permet de créer des interfaces en utilisant des fichiers CSS (QSS dans le contexte Qt). Elle est assez simple à mettre en place, mais propose des interfaces classiques et moins modernes en comparaison avec des solutions plus récentes. Il n'y a pas de séparation nette entre le backend et le frontend, ce qui peut rendre la maintenance plus complexe à long terme.



- **Python backend + QML** : Cette combinaison permet de séparer clairement le frontend du backend grâce à QML, un langage orienté interface utilisateur. Il permet de créer des designs modernes et fluides. La mise en œuvre est plus complexe que l'utilisation de QSS, mais offre une plus grande flexibilité et un rendu plus dynamique.

Critères de comparaison :

- Séparation du frontend et backend (modèle MVC)
- Modernité et fluidité
- Compatibilité avec PyQt
- Simplicité d'implémentation



Comparaison des solutions :

Critères	Qt + QSS	Python backend + QML
Séparation backend/frontend		
Modernité et fluidité		
Compatibilité avec PyQt		
Simplicité d'implémentation		

Conclusion :

Nous utiliserons **Python backend + QML** pour sa flexibilité et modernité car nous voulons une application complète, belle et fonctionnelle.

3. Interprétation des prompts (détection d'objets)

Introduction :

L'objectif est de choisir le modèle de réseau de neurones le plus adapté pour la détection d'objets à partir de prompts textuels dans des images et vidéos.

Description des solutions :

- **YOLO World** : Ce modèle est largement reconnu pour sa rapidité et sa précision dans la détection d'objets. Il est facile à personnaliser et adapté pour une détection en temps réel, ce qui en fait un choix de premier plan pour les systèmes interactifs.
- **Faster R-CNN** : Ce modèle est très précis, mais beaucoup plus lent en traitement. Il est recommandé pour des tâches nécessitant une haute précision sans contrainte de temps réel.
- **SSD (Single Shot Detection)** : Il se situe entre YOLO et Faster R-CNN en termes de rapidité et de précision. Cependant, il est moins flexible et moins utilisé aujourd'hui.



Critères de comparaison :

- Précision de détection
- Performance en temps réel
- Flexibilité des hyperparamètres
- Documentation et support



Comparaison des solutions :

Critères	YOLO World	Faster R-CNN	SSD (Single Shot Detection)
Précision de détection	Yellow	Green	Orange
Vitesse	Green	Red	Orange
Personnalisation	Yellow	Yellow	Orange
Documentation/ support	Green	Yellow	Orange
Gestion des prompts	Green	Red	Red

Conclusion :

Nous utiliserons **YOLO World** pour l'application, mais nous gardons en tête les autres modèles qui pourraient être intégrés dans le mode expert.

4. Reconnaissance vocale (prompts multimodaux)

Introduction :

L'application doit permettre aux utilisateurs de saisir des commandes vocales, converties en texte, pour améliorer l'interaction avec la détection d'objets.

Description des solutions :

- **Whisper** : Modèle open-source qui offre une précision très élevée pour la reconnaissance vocale, y compris pour des accents variés. Fonctionne localement sans nécessiter de connexion internet, ce qui est un grand avantage.



- **Google Speech Recognition** : Très précis également, mais nécessite une connexion Internet pour fonctionner. Le coût peut augmenter en cas d'utilisation intensive.



- **CMU Sphinx** : Moins précis que Whisper ou Google Speech Recognition, mais très léger et open-source. C'est une option intéressante si la consommation de ressources est une contrainte.
- **Snowboy** : Spécialisé dans la détection de mots-clés. Très léger, mais limité à des scénarios simples de détection.

Critères de comparaison :

- Précision de la reconnaissance
- Vitesse de traitement
- Compatibilité avec Python
- Licence et coût



Comparaison des solutions :

Critères	Whisper	Google Speech Recognition	CMU Sphinx	Snowboy
Précision				
Vitesse de traitement				
Compatibilité				
Licence/coût				

Conclusion :

Nous utiliserons **Whisper**, car il permet l'interprétation en local, est open-source et est compatible avec Python.

5. Base de données (Encyclopédie + Historique)

Explication :

L'application doit stocker des données, d'un historique ainsi qu'une encyclopédie d'informations sur les objets détectés. Elle devra donc pouvoir être modifiée et utilisée en continu lorsque l'application sera en cours d'utilisation.

Description des solutions :

- **MySQL** : Solution robuste pour gérer de grandes quantités de données. Elle est idéale pour des applications nécessitant un support multi-utilisateur, mais demande un serveur et une configuration plus complexe.



- **SQLite** : Légère et facile à utiliser, cette base de données est parfaite pour des applications locales ou à petite échelle. Elle ne nécessite pas de serveur et est très simple à maintenir, mais elle est limitée en termes de gestion de gros volumes de données.



Critères de comparaison :

- Facilité d'utilisation et configuration
- Capacité à gérer des volumes de données variables
- Performances en lecture/écriture
- Maintenance et sauvegarde
- Licence et coût



Comparaison des solutions :

Critères	MySQL	SQLite3
Facilité d'utilisation		
Gestion de grandes données		
Support multi-utilisateur		
Maintenance		
Licence		

Conclusion :

Nous utiliserons **SQLite3** pour notre application, car la base de donnée sera simple et légère ce qui permettra un hébergement en local.

6. Transformation de l'application en fichier exécutable

Introduction :

Les utilisateurs pourront installer l'application simplement en installant un fichier exécutable, sans devoir avoir une installation de python sur l'ordinateur, ni devoir s'occuper des dépendances. Car nous voulons que l'application soit accessible au plus grand nombre.

Description des solutions :

- **Cx Freeze** : Permet de générer des fichiers exécutables relativement légers, mais il est plus difficile à utiliser pour les débutants. Compatible avec Windows, Linux et macOS.
- **PyInstaller** : Très simple d'utilisation et offre une compatibilité étendue avec plusieurs OS. Cependant, les fichiers exécutables générés sont plus volumineux car ils incluent les bibliothèques nécessaires au programme.

Critères de comparaison :

- Facilité d'utilisation
- Taille des fichiers .exe retournés
- Compatibilité avec les OS



Comparaison des solutions :

Critères	Cx Freeze	PyInstaller
Facilité d'utilisation		
Taille des fichiers .exe retournés		
Compatibilité avec les OS		

Conclusion :

Nous utiliserons **PyInstaller** pour notre application, car le logiciel est plus simple d'utilisation et compatible avec les trois principaux systèmes d'exploitation.

Conclusion

Après une analyse approfondie des différentes technologies disponibles pour chaque besoin du projet, voici les outils sélectionnés :

1. Conception de l'interface graphique :

Qt Design Studio a été retenu pour son support du QML, son interface complète et fonctionnelle, qui permet la création d'interfaces complexes.

2. Interface utilisateur :

Python backend avec QML a été retenu pour assurer une interface moderne, fluide et réactive. Ce choix garantit une meilleure séparation entre le frontend et le backend, facilitant ainsi la maintenance et les futurs ajouts de fonctionnalités.

3. Interprétation des prompts (détection d'objets) :

Le modèle **YOLO World** a été sélectionné pour la détection d'objets en raison de sa rapidité et de sa précision. Il offre une personnalisation facile et des performances adaptées aux besoins de l'application, permettant une détection en temps réel sur des images et des vidéos.

4. Reconnaissance vocale :

Whisper, un modèle de reconnaissance vocale open-source, a été choisi pour son excellente précision et sa capacité à gérer des langues variées et des accents, ainsi que pour son fonctionnement en local.

5. Base de données :

SQLite a été retenue pour la gestion des données (encyclopédie et historique) en raison de sa légèreté et de sa simplicité de mise en œuvre. C'est une solution idéale pour un projet avec des exigences simples en termes de gestion de données

6. Transformation de l'application en fichier exécutable :

PyInstaller a été retenu pour sa facilité d'utilisation et sa compatibilité avec les 3 systèmes d'exploitation principaux